November 2020

On-device Query Cache For Digital Assistant

Xin Li

Chen Zhang

On-device Query Cache For Digital Assistant

ABSTRACT

There may be failures or delays in query fulfillment when a digital assistant utilizes server-side query processing to interpret a query and provide an appropriate response. The delay can be worse in certain situations, e.g., a poor network connection when a user places queries while in a moving vehicle. This disclosure leverages the observation that many queries are repeated verbatim by users. The described techniques, with user permission, utilize an on-device query cache to reduce latency in query fulfillment. With user permission, spoken commands are matched with prior queries stored in the on-device cache and if a match is found, the query is fulfilled locally. In the absence of a match, server-side query processing is performed, and the query is added to the client-device cache. The cache is updated based on user-permitted factors such as query recency, query frequency, etc.

KEYWORDS

- Digital assistant
- Virtual assistant
- In-car query
- Query intent
- Query cache
- Infotainment system

BACKGROUND

Interpreting and responding to communication queries, e.g., “Call X”; “Send a message to X”; etc., issued to a digital assistant involves many repeated steps which can take substantial processing time and can involve a remote server (if server-side processing is permitted by the

user). For example, to act on the spoken command "call mom," issued to a digital assistant provided via a client device, e.g., an in-car infotainment system, etc.), the operations may involve:

- User pairs their phone with in-car infotainment system and allows the system to access the user's contacts; when the phone unpairs from the in-car infotainment system, all of the user's contacts information is deleted automatically from the system
- Send speech data including the spoken command and associated context to a remote server for recognition
- Determine the query intent at the remote server based on the received query and context, and available user data such as the user's contacts ("Mom") and relationship ("mother")
- Trigger one or more features on the remote server based on the determined query intent, including "calling a contact"
- Choose a particular feature ("calling a contact") based on a model that takes into account the user's action history and determine the appropriate contact to call ("Mom")
- Match local contacts on the client device with contacts stored on the server, and place the call from the client device (query fulfillment) upon confirming the match

Performing these operations that involve both client and server-side processing can introduce significant latency.

Also, communication queries to a digital assistant are often repeated queries. For example, if a user initiates a call to their mother via a digital assistant with the phrase "call mom," they are likely to use the same phrase each time when calling their mother. Also, a large proportion of users typically place calls to a relatively small number of contacts using voice commands issued to a digital assistant.

DESCRIPTION

This disclosure describes the caching of client-side queries and associated actions to detect repeat executions of the same query and perform the associated action directly on the client device, without the latency associated with sending information to a remote server and receiving the query interpretation. The caching is performed with specific user permission; if the user denies permission, no caching is done and the default, user-permitted mechanisms for query fulfillment are utilized. The techniques can be implemented in an on-device digital assistant application, e.g., in an in-vehicle infotainment system, or other device.

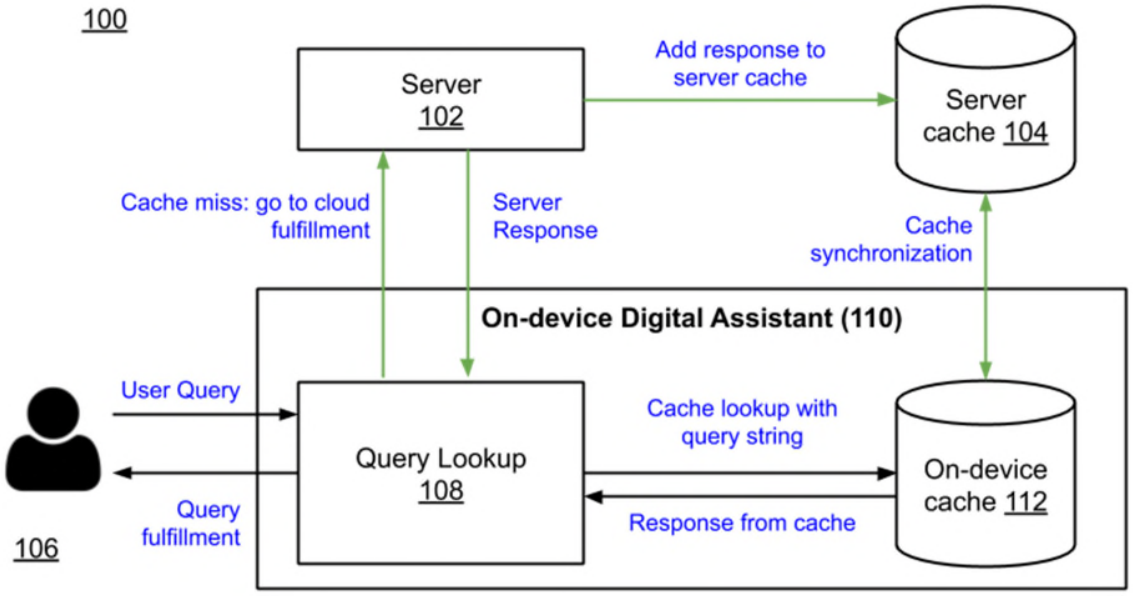

Fig. 1: Example framework of using on-device cache for optimizing virtual assistant communication queries in vehicles.

Fig. 1 shows an example framework (100) of using an on-device cache to reduce latency of providing a response to a user query to a digital assistant (110) on a client device. A user (106) provides a query, e.g., a spoken command “Call mom” to the digital assistant. With user permission, a query lookup (108) is performed by looking up the on-device cache (112) with the query string, e.g., the text “call mom.” If the query is in the on-device cache, query fulfillment is

achieved locally by the appropriate action (e.g., initiate a call to the contact “Mom”) being performed by the digital assistant. This flow is illustrated by black arrows in Fig. 1.

When interpreting the spoken command, it is determined that the command is to place a call to the contact named in the query. Contact name resolution is then attempted locally on-device by accessing the locally stored contact list, with user permission, such that names that are locally stored and are close matches to the spoken name in the query are preferred. For example, if the user says “Call [henri]” and the closest name in the locally stored contact list is "Hengyu", then the call query is interpreted as call to Hengyu and executed accordingly; f the closest locally stored name is "Harry", then the call is directed to Harry. If no local contact is found that matches the query with a sufficient confidence level, it is determined that there is no match in the on-device cache.

When the query lookup fails - there is a cache miss, since the query doesn’t have a match in the on-device cache - the query is sent to a server (102), if permitted by the user. This flow is illustrated by green arrows in Fig. 1. Query interpretation is performed at the server and a response is provided to the client device, which then performs query fulfillment accordingly. Additionally, with user permission, the query and response is added to a server cache (104). The server cache is synchronized with the on-device cache.

In this manner, repeated or frequent queries are automatically added to an on-device cache and can be fulfilled locally, thus reducing latency. The cache can be indexed by the specific query string (“Call mom”) since repeat queries tend to be exact matches. The cache can be automatically updated to remove queries that haven’t occurred recently, changes in context (e.g., a different user is using the client device, contact updated in the address book, different location, etc.)

The described techniques can be implemented for local query fulfillment for queries with communication intent (e.g., calling, sending a message, etc.) or other suitable queries. Client devices such as vehicle infotainment systems (e.g., that can be paired with a phone or other device) can incorporate these techniques. The reduction in latency and the ability to serve queries locally (which works even in case of network failure) improves the user experience of interaction with the digital assistant.

Further to the descriptions above, a user is provided with controls allowing the user to make an election as to both if and when systems, programs, or features described herein may enable collection of user information (e.g., a user's spoken commands or queries, a user's contacts, a user's current location), and if the user is sent content or communications from a server. In addition, certain data are treated in one or more ways before it is stored or used, so that personally identifiable information is removed. For example, a user's identity is treated so that no personally identifiable information can be determined for the user. Thus, the user has control over what information is collected about the user, how that information is used, and what information is provided to the user.

CONCLUSION

This disclosure leverages the observation that many queries are repeated verbatim by users. The described techniques, with user permission, utilize an on-device query cache to reduce latency in query fulfillment. With user permission, spoken commands are matched with prior queries stored in the on-device cache and if a match is found, the query is fulfilled locally. In the absence of a match, server-side query processing is performed, and the query is added to the client-device cache. The cache is updated based on user-permitted factors such as query recency, query frequency, etc.