December 2021

Privacy-Preserving Application Programming Interface for Discovery of Features

Eugenio Marchiori

Privacy-Preserving Application Programming Interface for Discovery of Features

Abstract:

This publication describes methods, implemented through an application programming interface (API) on a computing device, for handling application feature discovery requests while preserving user privacy. To preserve user privacy, responsive to receiving a request (call) from an application (a caller) relating to the availability of a feature, the API provides a response indicating that the feature is not available and that it will query a service to make the feature available (e.g., download a module). The response is provided by the API regardless of whether or not the feature is currently available on the device. In aspects, the API then starts a timer and upon expiration of the timer indicates to the caller that the feature is now available, mimicking a delay associated with downloading and installing the feature.

Keywords:

Application programming interface, API, user privacy, on-device storage, feature discovery

Background:

Some application programming interfaces (APIs) hold state information (e.g., modules, user data, settings, preferences) that is potentially shared across different applications making API requests (callers). For example, an on-device speech API may be configured to download and utilize separate language models (state information) for each language selected by a user. In order to conserve on-device storage space, the API may share the state information between multiple applications. For example, a word processing application and a social media application may both utilize French and English language models installed on the computing device.

Disclosure of the presence (or absence) of particular state information held by an API may have privacy implications (e.g., fingerprinting). For example, by knowing what language models are installed on a computing device, an application may be able to determine what languages in which the user is interested or speaks. A traditional approach to a privacy-protecting API is for the API, upon receiving a call to discover a feature, to reveal all possible features to the caller. With user input, the caller would select a specific feature (e.g., language model), which the API would then request for download from an external program or web server, with multiple copies of the feature installed on the computing device (one copy for each caller). While this approach conserves the privacy of the user by not permitting the discovery of features with privacy concerns, it does not conserve on-device storage space since the API re-downloads the feature (e.g., language model) and stores it every time a new caller queries the API regarding the availability of the feature.

Description:

This publication describes methods, implemented through an API on a computing device, for handling application feature discovery requests while preserving user privacy. To preserve user privacy, responsive to receiving a request (call) from an application (a caller) relating to the availability of a feature, the API provides a response indicating that the feature is not available and that it will query a service to make the feature available (e.g., download a module). To protect the user's privacy, this response is provided by the API regardless of whether or not the feature is currently available on the device.

To conserve on-device storage space, any time a caller makes a request for discovery of a feature (e.g., a user’s preferred language model, current location, social activity), the API should remember how it responded, for example, by storing this information in a secure, on-device keystore, software register, or other secure storage medium. If a subsequent caller requests discovery of the same feature, the API should first reply that the feature is not available, then provide the feature in an opaque fashion (caller is ignorant to all previous requests) by potentially waiting an arbitrary amount of time before providing the subsequent caller with the requested feature.

Protecting user privacy and conserving on-device storage space are two objectives any API that manages application feature discovery requests may fulfill. Disclosed in this publication is an on-device API (e.g., speech API) that always behaves naively towards callers but remembers previous requests. This way, every caller to the API remains ignorant of previous requests and the API conserves on-device storage space by remembering previous requests. To demonstrate this concept, a detailed example is used in which there are two applications making requests to the on-device speech API, application A (App A) and application B (App B). Both callers, App A and App B, would like to use a language model for language N (Lang N) to provide speech recognition to the user.

In step one, App A queries the API to see if the Lang N model is available (already installed on the computing device). Regardless of whether or not the Lang N model is available, the API replies that the Lang N model is not available, and App A learns nothing about the user. In step two, the API downloads the Lang N model from an external program or web server, potentially storing the model (feature) in a secure, on-device keystore or similar storage solution. In step three, App A queries the API again to see if the Lang N model is available. The API replies that it is available to App A and provides App A with the model. In step four, App B queries the API to see if the Lang N model is available. The API replies that it is not available, even though it was previously requested by App A and downloaded from an external program or web server. Through the API's reply, App B learns nothing about the user. In step five, the API pretends to request that the Lang N model is downloaded, potentially waiting for an arbitrary amount of time (e.g., sets a timer). In step six, App B queries the API again to see if the Lang N model is available. The API replies that it is available to App B and provides App B with the model. A flowchart illustrating how an API may handle any application making a request for a feature (e.g., language model) is shown in Fig. 1.

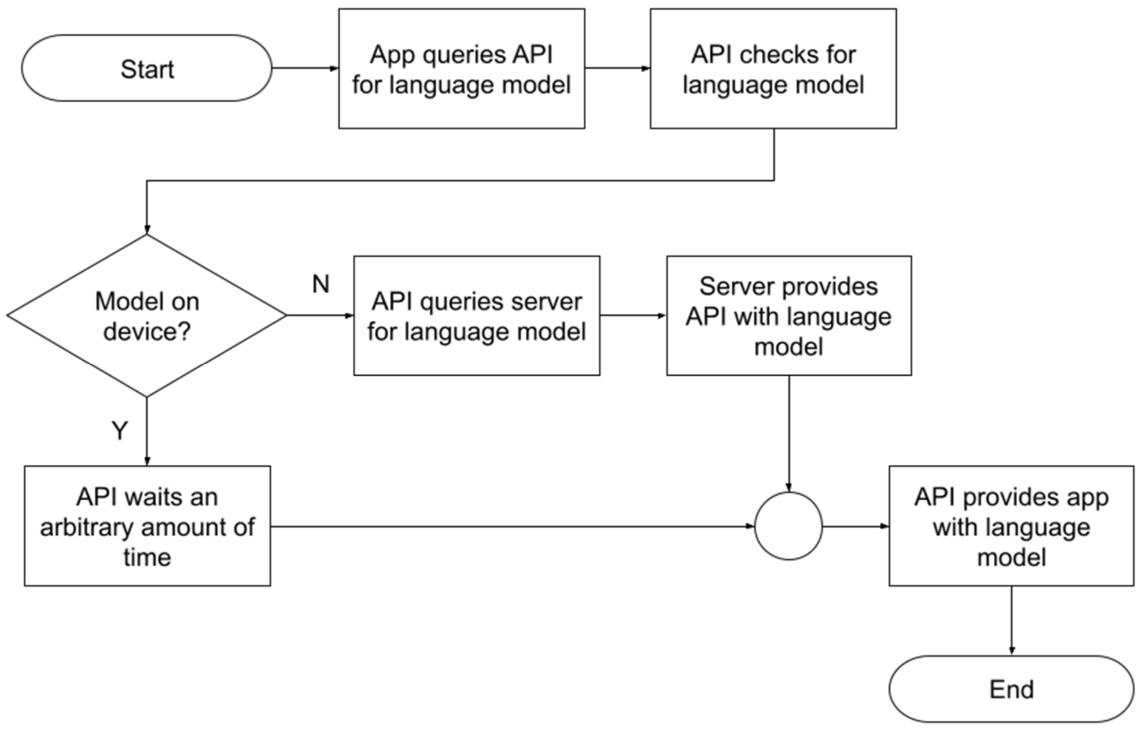
Start
App queries API for language model
API checks for language model
Model on device?
N
API queries server for language model
Server provides API with language model
Y
API waits an arbitrary amount of time
API provides app with language model
End

Fig. 1. Flowchart of on-device speech API decision tree once queried for a language model.

An additional step that could be included is for the API to remember how long it took to download the Lang N model. Rather than waiting an arbitrary amount of time after App B queries the API, the API could wait the exact amount of time it took to download the Lang N model initially to indicate to an application that the feature is available.

Further to the above description, a user may be provided with controls to make an election as to both if and when a caller described herein may collect user information through the request of a feature (e.g., a user's language model), and if the user is sent content and/or communications from an external program or web server. In addition, certain data may be treated in one or more ways before it is stored and/or used, so that personally identifiable information is removed. For example, a user's identity may be treated so that no personally identifiable information can be determined for the user. In another example, where location information is obtained, a user's geographic location may be generalized to a city, ZIP code, or state, so that the specific location of the user cannot be determined. Thus, the user may have control over what information is collected, how that information is used, and what information is provided to the user.

References:

[1] Patent Publication: US20140283132A1. Computing Application Security and Data Settings Overrides. Priority Date: March 12, 2013.

[2] Patent Publication: US20160048695A1. Method of Preventing Access to Sensitive Data of a Computing Device. Priority Date: March 28, 2013.

[3] Dutton, Sam. "Digging into the Privacy Sandbox." web.dev, Published: April 8, 2020. https://web.dev/digging-into-the-privacy-sandbox/.