Technical Disclosure Commons

Defensive Publications Series

March 23, 2017

Smart Button Actions On Mobile Devices

Thomas Deselaers

Daniel Keysers

Victor Carbune

SMART BUTTON ACTIONS ON MOBILE DEVICES

ABSTRACT

A system and method are disclosed that trigger a smart response to a button action when two or more running applications have applicable actions on a mobile device. The system includes a machine learning algorithm (MLA) built into the operating system that analyzes and learns from user actions. When the system detects a hardware button press, it retrieves possible actions/events that may be triggered for the current app and other running apps and processes. The actions are evaluated by computing a combination of scores involving machine-learned, rule-based scores provided by the apps and real-time signals such as location, time, user activity, etc. The system then triggers an action that has the highest score surpassing a predetermined threshold. The disclosed method would provide improved user experience, and also new, useful button actions that previously were not available to the user.

BACKGROUND

Currently, smartphones typically have three or four physical buttons and occasionally an extra button is connected to the phone, e.g. when a headset with a button is connected or when a stylus with a button is connected. These buttons generally have very fixed use-cases, e.g. power on, volume up/down and home. Some of the other buttons, e.g. those on a headset or stylus, often don't do anything at all in many contexts.

Some eBook apps use the volume buttons to flip through pages, which is a good example of a ‘smarter’ use. A problem that arises here is what happens when you are receiving a call while reading a book. Currently there is no way to decide whether pressing the volume down button would reduce the volume or switch to the next page.

Apps currently have to implement special handlers in order to provide functionality for these buttons. For the remaining apps that might not use external buttons, when the user clicks these buttons, nothing happens.

DESCRIPTION

A system and method are disclosed that trigger a smart response to a button action when two or more running applications have applicable actions on a mobile device. The system may include a mobile device with an operating system to which an external button is connected. The operating system of the mobile device may include a machine learning algorithm (MLA).

The system is built into the operating system or a similar basic layer of the program stack that is running on the device in order to be able to reason about and modify user interactions like button presses using the algorithm. In addition, the system will require the apps that want to participate in the process to expose an API that would implement the method. Alternatively, the OS/system could determine which events have handlers installed in the given app(s) that may be triggered. The machine learning algorithm would recognize that an event is not handled by an app, or that the app/process that would handle the event is not the right app to do so. The system would then determine the most probable user action within the currently running apps/processes and learn from user actions.

The method is implemented on a mobile device as shown in FIG. 1. In the first step, the system detects a hardware button press. The button generally does not have an action assigned to it (i.e. it is not handled by the current app and the only default action is either unspecified or an often undesired default action like ‘increase ringer volume’ on ‘volume up’). The system retrieves the possible action/events that may be triggered for the current app and other running apps and processes. In the next step, the best matching action among the actions is determined from the current or running apps and processes. This determination may include computing a combination of scores from several sub-components. The several sub-components may involve machine-learned scores, rule-based scores , and scores provided by the apps. The scores may also depend on signals that are available to the system e.g. time of day, location, running apps, most recent apps used, past actions in the apps, etc. The algorithm determines if the evaluated score surpasses a required threshold. As shown in FIG. 1 if the evaluated score surpasses the required threshold, the highest scoring action is triggered, else no action is triggered. In some cases, the method may also implement a default action, if applicable.

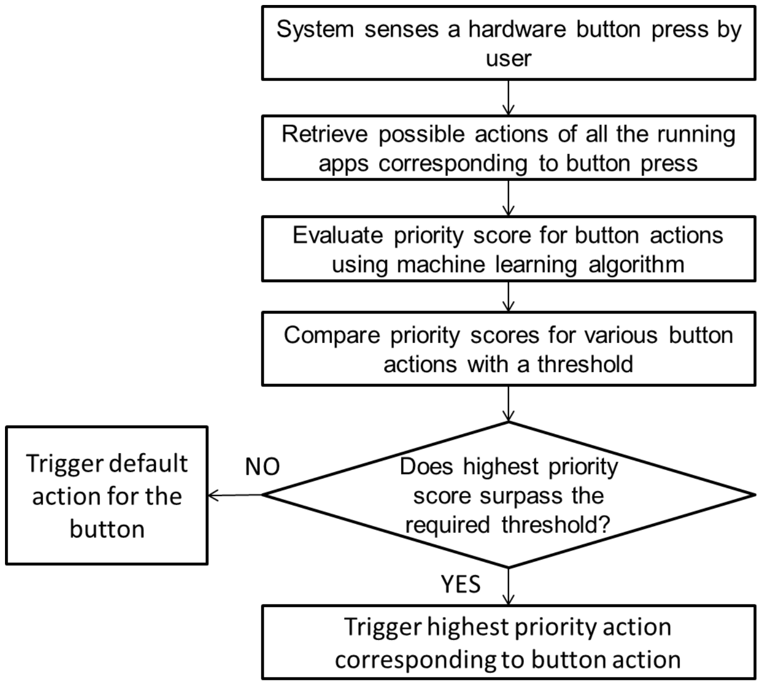
System senses a hardware button press by user
Retrieve possible actions of all the running apps corresponding to button press
Evaluate priority score for button actions using machine learning algorithm
Compare priority scores for various button actions with a threshold
Does highest priority score surpass the required threshold?
NO
Trigger default action for the button
YES
Trigger highest priority action corresponding to button action

FIG. 1: Method to trigger smart button actions on mobile devices

Additionally, the MLA would be able to learn from the user actions for a particular context, thereby improving the behavior over time. The feedback may be used in multiple ways: mapping in the particular context the actual action that would have been preferred (e.g. the action done by the user after undoing the automated action), and use as negative feedback that this action should not be done in that particular context in order to not select it next time, when a similar context appears. In the operating system context, the MLA would consider what apps are running, and a list of registered and unregistered actions thereof, detection of app-specific context such as text details, media details etc. There could also be detection of user specific context such as environment signals which includes location, time of day, calendar, previous actions and so on. The target of the machine learning algorithm is to score a set of actions that apps have registered and sort them by this threshold score.

The method of app prioritization could also be exposed as an API (application programming interface). The app implements a small API which evaluates and scores the most likely action that the app desires. The app may return an empty set of actions if they don't have any obvious next steps. The app may also indicate the action that corresponds to an undo of each of the actions performed. This action may be useful if the button pressed has an obvious opposite (e.g. volume up / down). In the case where there is the "opposite" action, it is useful feedback for the MLA when the user does something which triggered an action A, and then immediately undoes action A by doing the opposite. This tells the MLA that it predicted the wrong action A for the button press. The method of prioritization through API enhances the learning of the system. The app also exposes the API to trigger one of the suggested actions.

The machine learning model used may be a simple linear model (such as logistic regression), hidden Markov models or recurrent neural networks, depending on how the signals are combined. Another option would be a mixed model that combines sparse features with dense features to predict the score/likelihood of each individual action.

Also, as an alternative to an API, the OS may determine the events (button presses) installed by handlers that would react to a given event. The different actions may be hard coded in both the system and the running app.

The following are a few cases where the system and method finds application. When the user receives a call while he is reading a book, the action that should be carried out is prioritized by the system. If the ringer volume is not at its minimal volume the user would likely want to decrease the ringer volume. If the volume is already very low, the app (depending on what it learned from the user) could decide to reject the call.

In another example, a user may be using a web browser with no music playing. If the user clicks the volume up button, the ringer volume is not increased as the music app is inactive. The browser now exposes through the API that "back" is an option and that "forward" is it's undo, and the browser goes back. If the user is happy, the action ends. If the user immediately hits volume down, the browser goes forward again to undo the action. The system may now use this to learn that this was not the preferred action and goes on to implement the next probable alternative.

When the user is exercising, running outdoors, he is listening to music and the button on the headset with no assigned activity is pressed. There could be three possible actions: ask the run tracking app to give a current status, pause the music, open the microphone to allow for communicating with the speech enabled assistant app. Here, the system could trigger the first action. If the user clicks the button again, it shows that he is not interested in that action, but maybe wants to communicate with the assistant app.

The user is entering text using a stylus and handwriting in a (web) form. While in some cases the stylus-button would trigger e.g. an erase action, in a form, the button may better trigger a ‘go to next input field’ action (like hitting ‘tab’ on a keyboard).

The user is reading a book with an e-book app, while listening to music on a headset. Clicking the headset button may either pause the music or flip to the next page. The system may take into account the duration taken by the user to read a page and time when the page should be flipped. If the page has just been flipped, “pause” is the more likely preferred action.

If the user is watching a video that has no sound, the volume buttons could be used for fast forward/backward control in the video.

If the user is listening to music on a headset and triggers the volume down button, the system may disambiguate whether to pause the music entirely because someone else approached the user, or simply turn the music volume lower. The system may use microphone signals to detect directed external voice sounds.

The disclosed method would provide improved user experience. Another advantage is that the system provides new, useful button actions that previously were not available.