July 23, 2019

REVERTING ACCIDENTAL KEY TAPS USING CONTEXTUAL BANDITS

Victor Carbune

Alexandru Damian

REVERTING ACCIDENTAL KEY TAPS USING CONTEXTUAL BANDITS

ABSTRACT

A computing device (e.g., a mobile phone, camera, tablet computer, etc.) uses a contextual bandit machine-learning model (e.g., an artificial intelligence model) for reverting accidental user inputs. The computing device may execute an application that employs a user interface to facilitate human-machine interaction. The computing device may collect contextual user data and use a probabilistic model (e.g., a contextual bandit machine learning module) to analyze the collected contextual data to determine a confidence interval for specific user input. The application may receive an indication of the user input, e.g., tapping a particular icon within a graphical user interface (GUI) of the computing device to send to another user. The user may accidentally tap the wrong icon and unknowingly send an unintended communication to the other user. The contextual bandit, may, upon receiving and analyzing context data, determine, within a confidence interval, that the user likely did not intend to send the tapped icon. In this case, the computing device may display a message confirming the user's intention to send the selected icon. The contextual bandit updates the probabilistic model based on the user's response to improve the model's future decisions of whether to query the user about suspected incorrect user inputs.

DESCRIPTION

A user may interact with applications executing on a computing device (e.g., a mobile phone, tablet computer, smart phone, desktop computer, or the like). In some examples, a computing device may include or communicate with a touch-sensitive display that may enable a user to interact with applications executing on the computing device. Some applications may provide functionality through receiving indications of various user inputs, such as key taps. Due to the physical size of the screen and a user's ability to carefully, accurately tap different icons, the user may occasionally tap something other than their intended target, and an unintended message is sent (e.g., a wrong gif or sticker in a messaging application) or an unintended action is performed (e.g., a wrong friend request sent on a social media application).

The user should be provided with the ability to undo most of their actions determined to be likely accidental, without affecting the user experience for intended actions. This disclosure provides techniques that may mitigate some of the unintended consequences of incorrect taps, using a probabilistic model built to indicate how likely a certain action was actually desired to be performed by the user based on context such as the previous actions, context of the application where the action is being executed, and the action itself.

This disclosure describes a machine-learning based contextual bandit model that, given a state as input (contextual signals), outputs an indication of whether an action should be executed immediately or delayed by requesting the user to confirm. These two options may be referred to as two "arms" of a multi-arm contextual bandit reward model. Unlike supervised machine learning, which requests a user to confirm after every user action, a contextual bandit model's unsupervised learning process attempts to optimize its output based on previous user responses to the user confirmation requests. These previous user responses may have been based on previous user contexts. For example, for a particular action the contextual bandit model may be used to determine, based on previous confirmations for the particular action, that the user has a certain probability to confirm the particular action. If the certain probability satisfies a threshold, the contextual bandit will output a request to the user to confirm the action. That is, if it is more likely than not that the user selected an incorrect action, the contextual bandit requests a confirmation.

Two competing interests are at play in a contextual bandit's learning process: to collect preferences by requesting the user to confirm, and to determine whether to execute the action immediately. That is, to either gain information to make the contextual bandit more precise or decide based on less information. Although the contextual bandit may incorrectly request the user to confirm when the user intended the action (e.g., to attach a gif), the contextual bandit will have more information to include in a corpus for future decision making. For example, the user wishes to select an icon (e.g., from among twenty icons) within the application to send to a coworker; each of the twenty icons have an associated probability distribution. The contextual bandit may analyze the selected icon and any previous uses of the selected icon with respect to any contextual signals associated with previous uses of the selected icon. The contextual bandit may not make any inferences about any of the other nineteen icons from the selected icon. A probabilistic two-arm reward model, which depends on the selected icon and associated previous selected icon usage, determines whether the contextual bandit will send the request for confirmation.

Figure 1 illustrates a computing device having a user interface (UI) device, an output device, and a contextual bandit module. Example computing devices include mobile phones, tablets, digital cameras, laptops, gaming systems, e-book readers, televisions, wearable computing devices, or any other type of mobile or non-mobile computing device connected to a display that may transmit messages (e.g., emails, text, SMS, etc.). The example computing device may output a request asking a user to confirm executing an action (e.g., attaching a gif to an email). For instance, the computing device may collect data indicative of contextual signals, including application specific text, images, videos, screenshot data, actions taken by the user within a predetermined timeframe (e.g., within the past twenty milliseconds), etc.

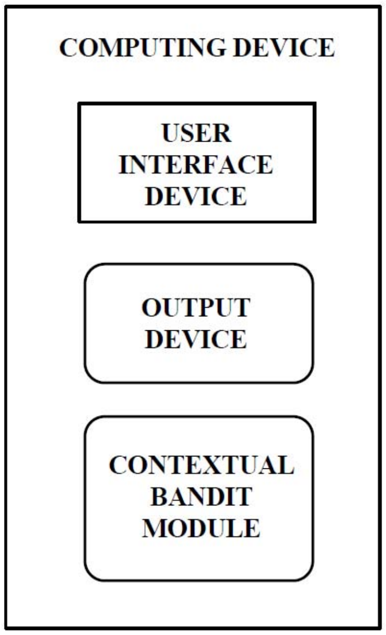

FIG. 1

The user interface device may function as an input device for the computing device, such as using a presence-sensitive input screen. The output device may, e.g., attach a gif the user selects to an email, text message, etc. The output device may function as an output (e.g., display) device using one or more display devices, such as a liquid crystal display (LCD), dot matrix display, light emitting diode (LED) display, organic light-emitting diode (OLED) display, e-ink, or similar monochrome or color display capable of outputting visible information to a user.

The user interface device may control a graphical user interface including determining what the user interface device presents and what information is exchanged between the user interface device and other applications or components (e.g., the output device and the contextual bandit module) of the computing device. For example, the user interface device may receive information from a component of computing device for generating a user interface and elements thereof. In response, the user interface device may output instructions and information, causing the graphical user interface to display an application. The contextual bandit may receive information from the user interface device in response to inputs detected at locations of a screen of the user interface device at which elements of the graphical user interface are displayed. The user interface device disseminates information about inputs detected by the graphical user interface to other components of the computing device for interpreting the inputs.

The user interface device may cause the graphical user interface to facilitate interaction between users and the computing device. For example, the user interface device may cause the graphical user interface to display content associated with the application, including icons that a user may select to include in the application (e.g., a correspondence application). Figure 2 illustrates the contextual bandit module's learning process.

In some examples, the computing device may receive user input to perform an action (e.g., to send an emoji with a message to a coworker). Upon receiving an indication of the action, the contextual bandit module may refer to a stored corpus of information regarding the particular emoji and all historical user responses to requests to confirm whether the user intended to send the particular emoji. The contextual bandit module may then determine, based on the emoji-corpus and the contextual signals, whether the user likely intends to include the emoji in the message. In response to the contextual bandit module determining it is likely that the user intends to send the emoji, the contextual bandit module causes computing device to send the message without asking the user to confirm. In response to the contextual bandit module determining it is unlikely that the user intends for that particular emoji to be included in the message, the contextual bandit module may cause the output device to output a request asking the user to confirm the intention to send the emoji. Indications of received responses by the user may be stored within the emoji-corpus for further use.

The contextual bandit module uses context features and a context model. As examples, the contextual signals that may form the input data (state) for the contextual bandit algorithm may include: application context including application specific content, such as text content for a communication application, video or audio for others; user screen context such as a screenshot of the current screen rendered to the user (for button size and other visual configuration); the last few screens in terms of a few milliseconds displayed to the user (indicating whether the user quickly switched to this current application, or if the user was already using the application), other metadata signals that might be relevant (such as time of day, general user features, and the like), etc. The computing device may extract content (e.g., input data such as user previous action) from other applications the user has recently used. For example, the computing device may extract user generated data such as text from a different application when the user is switching back and forth between the different application and current application to increase context.

As described herein, the computing device collects the contextual signals and the contextual bandit module analyzes the contextual signals only after receiving explicit authorization from the user to do so. After receiving such authorization, the computing device and the contextual bandit module may begin to collect and analyze, respectfully, the contextual signals, sequentially.

In some examples, the computing device may employ machine learning technology in the contextual bandit module and may include neural networks (not shown) that may ingest the contextual signals by using adequate building blocks. Relative to FIG. 1, to train the contextual bandit model, the computing device may deploy the contextual bandit model to perform an action (take an arm) that is not necessarily the best decision, but is required for collecting data (i.e. to determine the user's feedback with respect to particular contextual signals and a particular icon). In this scenario, the contextual bandit module taking the default action of executing on what the user has tapped incurs no cost, because it is the default action that the user selected.

From time to time, the contextual bandit module may block an action to be executed (picking the other arm) and first output a prompt to the user: whether they want to roll back (revert) that action and simply cancel the action if receiving an indication that the user has confirmed. For example, the user may tap a particular icon out of twenty possible icons to include in a message to a coworker. The contextual bandit module may have insufficient data pertaining to the particular icon, so there is uncertainty whether this user action was an intentional touch (e.g., the user intended to select the particular icon), or whether this user touch was unintended (e.g., an accidental key touch). However, if the contextual bandit module were to guess (i.e., take an arm) by requesting that the user confirm that the user wishes to select the particular arm, whether the contextual bandit was right to make the request, or not, the contextual bandit now has more information regarding the particular icon to make a more informed decision in the future. The contextual bandit module uses reinforcement learning to maximize a probabilistic reward (likelihood reward).

In some examples, the computing device implements a fast execution computing framework to support the contextual bandit module, running effectively as an additional layer just before confirming the user tap. This can be done by integrating the contextual bandit system at the operating system level of the computing device, having a lightweight model architecture (e.g., only a few layers) that is more easily deployed on devices with some hardware acceleration for neural nets.

The techniques described herein could be applied at the application level in some examples, instead of at the operating system level. The techniques can be used by touch-enabled devices that employ any touch-based operating framework. For example, any smart device with a touch screen capable of receiving a command from a user to include a feature (e.g., icon, gif, etc.) within an application may utilize the contextual bandit module. This disclosure provides a model, and may include Application Programming Interface for fast execution of the model.

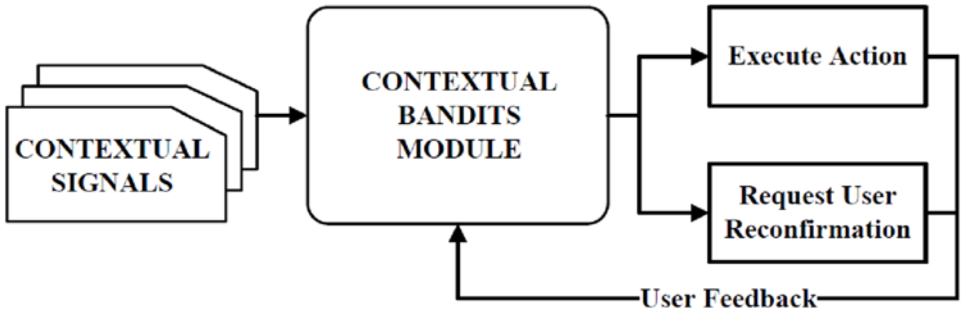

FIG. 2

As shown in FIG. 2, the contextual bandit module learning process is a self-adaptive process that includes receiving a set of contextual signals. The contextual bandit module may be based on a multi-arm bandit contextual model. For example, the contextual bandit module has two arms to choose from: take the default action (e.g., not interfere with the application transmitting an icon) or ask if the user has selected a correct action (e.g., selected an intended icon). The contextual bandit module learns when the contextual bandit module asks the user whether the user selected the correct action.

In some examples, the user may be using a specific application and the contextual signals may include application specific text, images, videos, screenshot data, actions taken by the user within a predetermined timeframe (e.g., within the past twenty milliseconds), etc. The computing device may receive user input to perform an action (e.g., to send an emoji with a message to a coworker). For example, the specific application may be an email application and the contextual signals may be text, images, or videos within an email that the user wishes to send to a coworker; text from other emails the user has sent within a predetermined amount of time; an email contact list, associated text of each address within the contact list, and a title designation for each contact; time of day or day of the week; and, any other application, other than the specific application, that the user has used within the past twenty seconds.

In some examples, the contextual bandit module may include a neural net module and the neural net module may ingest the contextual signals. In some examples, these contextual signals may provide the contextual bandit module with a baseline for an email the user is about to send to a coworker. In some examples, the contextual baseline may continuously receive contextual signals, or may begin receiving contextual signals upon a user action (e.g., sending an email).

In some examples, the email application may have a set of icons that a user may attach to an email. Each icon of the set of icons may have a corresponding probability distribution that, with respect to the received contextual signals, may indicate a likelihood the user intends to select a particular icon. With respect to each icon within the set of icons, the contextual bandit module may include an information corpus (not shown in FIG. 2) of past responses to requests for the user to confirm that the user intended to attach the particular icon to the email, along with associated contextual signals. After each response, each of these past responses may be transmitted along the user feedback path for storage within the corpus so that the contextual bandit module may learn more about the user's preferences.

In some examples, because of a small touch-screen size of the computing device, the user may have accidently touched on the frowning face area within the graphical user interface when the user was intending to select the smiley face. This may cause confusion when the coworker opens up the email and, after reading text indicating a positive outlook, sees the frowning face. So, prior to sending the email, the contextual bandit module may determine whether the action of attaching the frowning face icon to the email should be executed immediately (default) or be delayed by requesting the user to confirm, according to FIG. 2. If the contextual bandit module determines the default action is preferred, then the contextual bandit module will not collect any information to be used in future decision making. If the contextual bandit module requests the user to confirm they intend to send the frowny face, and the user indicates this was not intended, the response can be used to help train the contextual bandit module, and the user can now select, and then attach to the email, the intended smiley face icon. The user response may then be transmitted along the user feedback path to the corpus within the contextual bandit module for use in future decision making.

Further to the descriptions above, a user may be 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., information about a user's social network, social actions, or screenshots of the user's screen), and if the user is sent content or communications from a server. In addition, certain data may be 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 may be treated so that no personally identifiable information can be determined for the user, or a user's geographic location may be generalized where location information is obtained (such as to a city, ZIP code, or state level), so that a particular location of a user cannot be determined. Thus, the user may have control over what information is collected about the user, how that information is used, and what information is provided to the user.