Technical Disclosure Commons

Defensive Publications Series

November 27, 2017

API for learning and predicting user interactions

Sandro Feuz

Victor Carbune

API for learning and predicting user interactions

ABSTRACT

Software applications, e.g., mobile apps, at times require multiple user interactions to fulfill certain actions. This disclosure describes an API that returns a predicted user action, e.g., selection of an option in a software application, in response to a query by the application. The prediction of user action is generated using one or more trained machine learning models. The models are trained, with user permission and expressed consent, on prior user interactions with various apps. When users permit, other contextual factors, e.g., data from device sensors, other apps that are running, operating system data, etc. can also be used as inputs for the trained models. The requesting application can present the predicted action as a default setting, or can automatically use the prediction as the user selection. In this manner, the techniques can enable apps to reduce user interaction, e.g., the number of clicks/taps to complete an action.

KEYWORDS

- user interaction
- user behavior
- anticipatory design
- prediction model
- prediction API

BACKGROUND

Applications have different processes or interaction flows that require user inputs at several steps that lead to completion of user requested tasks. Some examples are:

- in an email application, the steps a user takes to search for, find, and pick a specific mail message from the list of messages;

- in an e-commerce application, options a user selects to fulfill an order, e.g., confirming payment method and shipping/delivery address from pre-filled options that originate from earlier attempts or stored user settings;
- in a maps application, selecting the mode of transport according to current context; etc.

From a user perspective, such flows require several user inputs, e.g., clicks/taps for selecting and completing an action, and details for alternatives to complete the requested action. To reduce the number of clicks/taps and improve such flows, app developers either rely on a handcrafted set of heuristics, or train a specifically tuned machine learning model for the particular app. However, such approaches do not scale across multiple apps that a user makes use of and the many actions/options within each app.

DESCRIPTION

This disclosure describes an API that returns a predicted user action, e.g., selection of an option in a software application, in response to a query by the application. The prediction of user action is generated using one or more trained machine learning models. The models are trained in an automated and largely unsupervised manner. In some instances, supervised training is also used, e.g., when users provide permission for use of feedback signals from an app that makes use of the API. The training data includes, with user permission and express consent, prior user interactions with various apps. The API can be used by any application, with little developer effort and cost.

Post training, the functionality offered by the API enables reduction in the number of manual steps required to fulfill an app action by enabling apps to predict user actions in different contexts, when user provides permission for use of the API. The model computes a confidence score for each action and the action with highest score is recommended to the app.

In response to a prediction, apps can prompt users to confirm or reject the suggestion. Alternatively, with user permission and express consent, apps can automatically accept the prediction if the confidence level is high. For example, an app can execute a query with the API to determine a default user setting for the app in a certain context. In response, the app receives an option (e.g., associated with a high confidence score) and either prompts the user to confirm the option, or with prior user permission, automatically uses the option for the default user setting. If the confidence level is low, e.g., as shown in the example pseudo-code of an API call of Fig. 1, users are prompted to take action and the user action is used to provide feedback to the API, if users permit such feedback.

```
prediction, confidence = Api.GetPrediction(...)
if(confidence is low)
   {prompt user to take action;
   record action via the API: id, action, set of possible actions}
Else
   {obtain user confirmation of prediction;}
```

Fig. 1: An example API call and user interaction based on prediction returned by API

The API is structured such that it is agnostic to the code of the app and can be integrated with different apps. The API uses one or more trained machine learning models to generate the prediction. With user permission, the machine learning models are trained by observing user actions in different contexts while using various apps. The user is provided with options to select the actions that are used for training, exclude one or more apps and/or actions from use as training data, and turn off the use of user data for such training.

When an app provides context for an action, the API can leverage such context. However, predictions can be made even when limited or no context is available. With user permission, the model can also learn from user feedback, e.g., user acceptance or rejection of a prediction, incorporating these as reward signals. In this manner, techniques of this disclosure train and implement a machine learning model for action prediction, and provide an API that apps can query. An app can use the action predicted by the API to provide a user interface with a default user action that is likely preferred by the user.

Techniques of this disclosure are illustrated by the following example. An API function call from an application causes the machine learning module to generate one or more predictions with associated confidence scores, based on various parameters. The top scoring prediction is provided to the application as a response. The below example illustrates the parameters that can be used for prediction:

```
pair<Prediction, Confidence> = predict(ActionID, Context, Options,
UserFeatures)
```

The Parameters `ActionID`, `Context`, `Options`, and `UserFeatures` are inputs, and `pair<Option, Confidence>` is the output. The output and input parameters are elaborated as follows.

- The output `pair<Option, Confidence>` comprises the predicted option or action, and a confidence level of the prediction.
- Input parameter `ActionID` uniquely identifies a specific action that is executed by the user. When permitted by the user, this parameter maps prior user behavior to actions, e.g., by matching past user decisions for this `ActionID`. For example, the `ActionID` could be a “mode_of_transportation_action” in a maps application.
- Input parameter `Options` represents available choices available to a user for `ActionID`. For example, in a maps application, `Options` may be “bus,” “car,” “bike,” “walk,” etc. In an e-commerce application, `Options` may be “home address,”, “work address,” etc.

The `Options` are in a format that is used by the application. The trained model returns one or more of these options to the app as the predicted user action (or selection) based on the context.

- Input parameter `Context` can be local to the app and/or incorporate other contextual factors, e.g., use of other apps, use of operating system, etc. when permitted by the user. Local context provided by the app, when made available with user permission and consent, comprises data accessible to the app in each situation, e.g., prior user actions, previous content viewed by user in the app, and contextual preferences indicated by a user. The other contextual factors, when made available with user permission and consent, can include user preferences, sensor data from the device (e.g., that the application may not have access to), data from the device display, other running apps on the device, location of the device, etc. The other contextual factors are specific to the user. Context is provided in an obfuscated or anonymized manner, based on user-specified preferences.
- Input parameter `UserFeatures` includes known and inferred user preferences. This data is encoded and used only upon specific user permission and express consent. For example, in a hiking app, a setting that a user prefers (e.g., easier hiking trails with moderate altitude changes) is a user-specific preference provided as input to the model.

While various factors are listed above as used by the API for action prediction, and can be provided as part of the API call, one or more of the factors can be excluded, e.g., based on user preferences. For example, `UserFeatures` are not provided if users do not provide preferences or decline permission to infer preferences. Optionally, an application registers explicitly with the API `ActionIDs` and corresponding `Options` that are available for each

`ActionID`. Alternatively, as the application is used, these values are dynamically determined based on user interaction and the context.

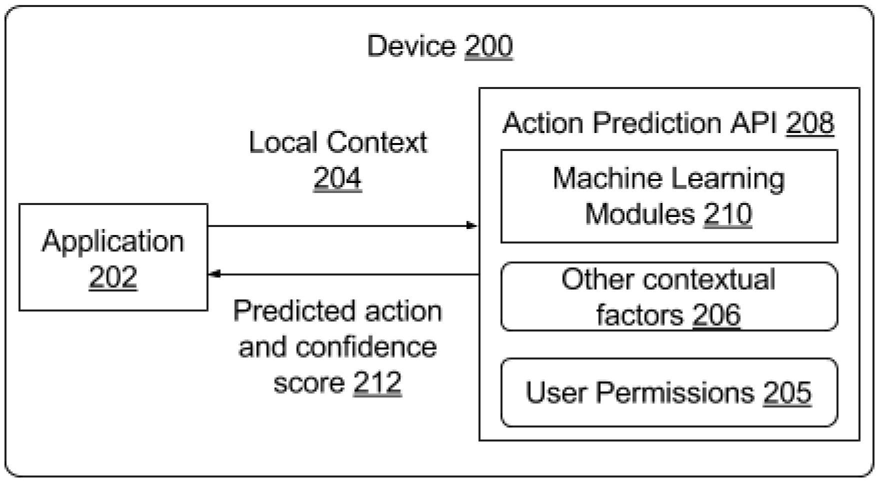
Device 200
Action Prediction API 208
Local Context 204
Machine Learning Modules 210
Application 202
Other contextual factors 206
Predicted action and confidence score 212
User Permissions 205

Fig. 2: API for user action prediction

Fig. 2 illustrates a user device (200) that implements an API (208) with machine learning modules (210). An application (202) makes a call to the API and provides local context (204). The API returns user action (212) and an associated confidence score, based on the local context and other contextual factors (206), based on user permissions (205).

In this manner, the described techniques transform a manual user selection of an option in app in any situation into an automatic prediction with the use of machine learning. The app developers themselves incur little or no cost and benefit from the reduction in user interaction with the app to provide input due to the automatic prediction. With user permission, user interactions with an app are provided training samples to the machine learning model. Alternatively, instead of implementing an API, the techniques can be incorporated within an app, or the app can use a simpler method such as maintaining a count or other statistic of user actions that is used to generate a prediction.

The machine learning modules can be implemented using several machine learning models jointly (e.g., one for each `ActionID`). Alternatively, the models can be trained individually and the results from each model combined to provide the prediction. For example, prediction of user action can be treated as a classification problem with different output layers, one for each `ActionID`, with the models trained jointly.

The model architecture can include, e.g., feed forward neural networks, recurrent neural networks, convolutional neural networks, support vector machines, random forests, boosted decision trees, etc. Separate neural networks may be used to process individual features. Reinforcement learning can be used, e.g., after initial training has been completed. For example, the user's agreement or disagreement with a predicted action can be incorporated as a reward signal to reinforce (prediction confirmed) or retrain (prediction rejected) the model, when users provide permission for such use.

Use of the API as described herein can result in a decrease in the number of user inputs required by apps. The API can be implemented as part of a device operating system. The techniques can be used in local, e.g., on-device apps, and web applications. The API can be incorporated with little cost to the application developer. Moreover, if users permit, a virtual assistant application can also make use of the described techniques.

In situations in which certain implementations discussed herein may collect or use personal information about users (e.g., user data, information about a user's social network, user's location and time at the location, user's biometric information, user's activities and demographic information), users are provided with one or more opportunities to control whether information is collected, whether the personal information is stored, whether the personal information is used, and how the information is collected about the user, stored and used. That is, the techniques discussed herein collect, store and/or use user personal information specifically upon receiving explicit authorization from the relevant users to do so.

For example, a user is provided with control over whether programs or features collect user information about that particular user or other users relevant to the program or feature. Each user for which personal information is to be collected is presented with one or more options to allow control over the information collection relevant to that user, to provide permission or authorization as to whether the information is collected and as to which portions of the information are to be collected. For example, users can be provided with one or more such control options over a communication network. 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. As one example, a user's identity may be treated so that no personally identifiable information can be determined. As another example, a user's geographic location may be generalized to a larger region so that the user's particular location cannot be determined.

CONCLUSION

This disclosure describes an API that returns a predicted user action, e.g., selection of an option in a software application, in response to a query by the application. The prediction of user action is generated using one or more trained machine learning models. The models are trained, with user permission and express consent, on prior user interactions with various apps. When users permit, other contextual factors, e.g., data from device sensors, other apps that are running, operating system data, etc. can also be used as inputs for the trained models. The requesting application can present the predicted action as a default setting, or can automatically use the prediction as the user selection. In this manner, the techniques can enable apps to reduce user interaction, e.g., the number of clicks/taps to complete an action.