December 10, 2018

Enhancing user interface accessibility using computer vision

Juhyun Lee

Enhancing user interface accessibility using computer vision

ABSTRACT

Accessibility features for vision-impaired users are typically provided by oral readouts of on-screen icons or other material. The task of transcribing in-app icons to text for text-to-speech conversion is typically left to the app developer. Icon-to-speech translation for apps from developers that are not conscious of the need to make their products accessible is often unsatisfactory. For example, a refresh button and a back button may both simply be transcribed as “button,” which is unhelpful to a vision-impaired user.

This disclosure uses computer vision techniques to automatically infer, and provide to a vision-impaired user, text corresponding to UI elements, e.g., icons, dropdown boxes, sliders, buttons, etc. The techniques are advantageously implemented on a platform over which apps operate, e.g., operating system, browser, etc., such that vision-impaired users can access UI elements even if the app developer did not transcribe in-app icons to text.

KEYWORDS

- Accessibility
- Computer vision
- User interface
- Icon-to-speech

BACKGROUND

Accessibility features for vision-impaired users are typically provided by oral readouts of on-screen icons or other material. The task of transcribing in-app icons to text for text-to-speech conversion is typically left to the app developer. Icon-to-speech translation for apps from developers that are not conscious of the need to make their products accessible is often

unsatisfactory. For example, a refresh button and a back button may both simply be transcribed as “button,” which is unhelpful to a vision-impaired user.

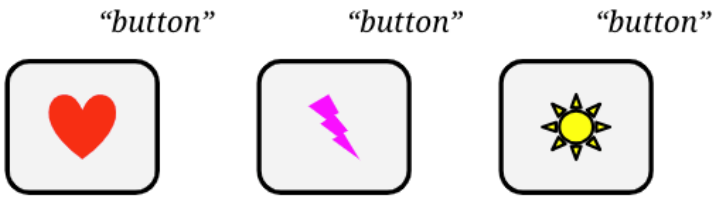

Fig. 1: Unsatisfactory icon-to-speech transcription

An example of unsatisfactory icon-to-speech transcription is illustrated in Fig. 1, which shows a variety of on-screen icons. The different buttons with distinct functions are simply transcribed as “button,” which is unhelpful to a vision-impaired user.

DESCRIPTION

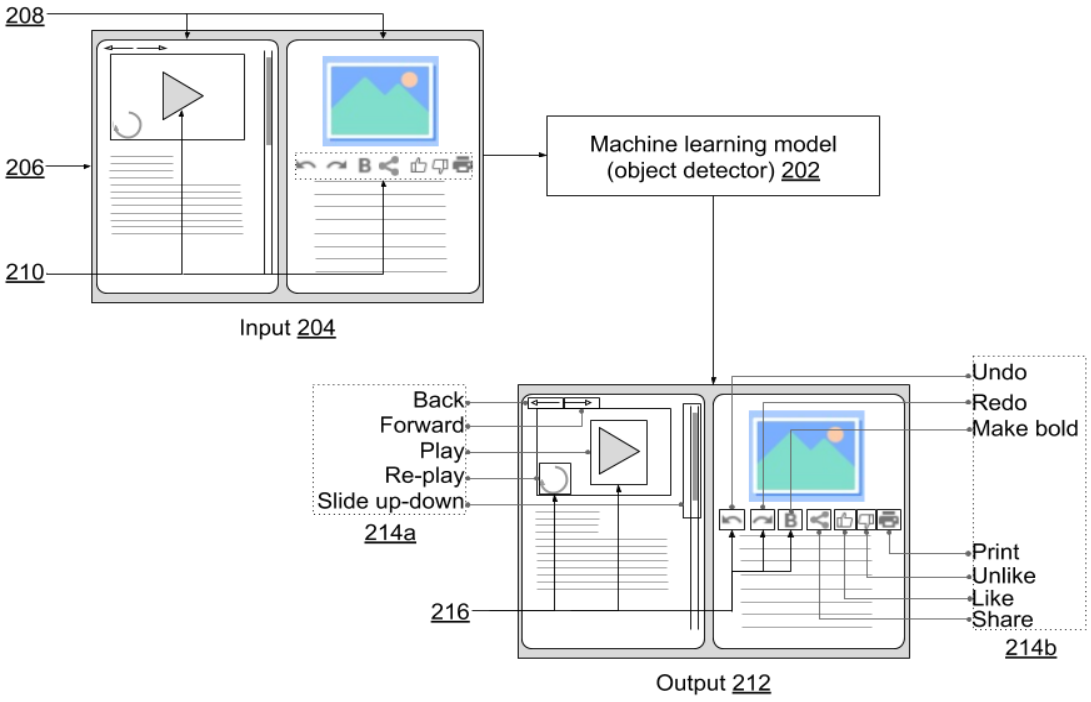

Fig. 2: Automatic transcription of on-screen UI elements using object detection

Fig. 2 illustrates automatic transcription of on-screen UI elements using a machine learning model, per techniques of this disclosure. The trained machine learning model (202),

which can be, e.g., an object detector, receives as input (204) a screen (206) comprising apps (208), each app including UI elements (210). The screen may be the rendered screen of a device such as a computer, mobile device, tablet, etc. The model returns as output (212) the screen with UI elements surrounded by bounding boxes (216). A bounding box is associated with a label (214a-b) indicative of the function of the bounded UI element. The label is converted to speech, e.g., when a user hovers over the UI element.

Training data for the machine learning model includes screenshots of various apps or web pages with labeled UI elements. The model can be trained in a supervised manner, e.g., labels for UI elements can be provided by a human labeler during the training phase. If human labeling is expensive, or if insufficient training examples are obtained using manual techniques, an automated, weakly-supervised training is possible, as follows. UI elements within web pages (HTML/CSS) are often accompanied by human readable labels. For example, the exit button in a page is typically labeled “exit” or “quit” in the HTML code. Training of the machine learning model can proceed by determining the boundaries of the UI element (e.g., from the operating system or the browser) and using the human-readable UI element descriptions as weak labels.

Using a machine learning model that bounds the UI element has the advantage of being able to assess a UI element in context. For example, the copy, cut, and paste buttons often appear together in applications; thus, there is high confidence in a button detected as “cut” if it is near buttons detected as “copy” and “paste.”

Alternative to providing the screen as input, a machine learning classifier can be constructed that takes in a single UI component and predicts its label. It is possible for the operating system (or other software such as a web browser) to determine the parameters of a UI element, e.g., size, type (e.g., slider bar, button, wheel, etc.), position, etc. The OS however lacks

information relating to the function of the UI element (e.g., “is the button an undo button or a redo button?”). A machine learning model that takes in a single UI element can be trained as follows. Given a set of labels - e.g., “volume up,” “volume down,” “undo,” “redo,” etc. - a training set for such a classifier is developed by searching the Internet for icons or images corresponding to the labels.

A hybrid approach, e.g., a combination of a machine learning model that accepts a screen as input (and provides labeled bounding-boxes as output) and a machine learning model that accepts a UI element as input (and provides UI element labels as output), can also be utilized. The hybrid approach can offer a performance improvement over its constituent models.

Accessibility can also be somewhat improved by simply reading out labels of UI components, if any, and if written in a language known to the user. For example, a “cut” button sometimes includes a human-readable label associated with it. The OS can simply read that label instead of learning to recognize the image of the UI element. The label can be machine translated prior to reading out.

The techniques are advantageously implemented on a platform on which apps operate and that renders third-party UI elements, e.g., operating systems, browsers, etc. If thus situated, the techniques described herein enable vision-impaired users to access UI elements even if the app developer did not transcribe in-app icons to text. In this manner, the user experience of vision-impaired users is improved without effort on part of app developers.

CONCLUSION

This disclosure uses computer vision techniques to automatically infer, and provide to a vision-impaired user, text corresponding to UI elements, e.g., icons, dropdown boxes, sliders, buttons, etc. The techniques are advantageously implemented on a platform over which apps

operate, e.g., operating system, browser, etc., such that vision-impaired users can access UI elements even if the app developer did not transcribe in-app icons to text.