Idiolect: A Reconfigurable Voice Coding Assistant

Breandan Considine
McGill University
bre@ndan.co

Nicholas Albion
Independent Developer
nalbion@yahoo.com

Xujie Si
University of Toronto
xsi@cs.utoronto.ca

Jin Guo
McGill University
jguo@cs.mcgill.ca

***Abstract*—This paper presents Idiolect, both an IDE plugin for voice coding and a novel approach to building bots that allows for users to define custom commands on-the-fly. Unlike traditional chatbots, Idiolect does not pretend to be an omnicient virtual assistant but rather a reconfigurable voice programming system that empowers users to create their own commands and actions dynamically, without rebuilding or restarting the application. We present a case study of integrating Idiolect [1] with the IntelliJ Platform, illustrate some example use cases, and offer some lessons learned during the tool's development.**

***Index Terms*—speech recognition, voice programming, bots**

I. Introduction

Humans are able to learn new words and phrases, and apply them in a variety of contexts relatively quickly. This is currently not the case for bots, which are often limited to a set of pre-defined commands and phrases and hinders usability. Users become frustrated when they are unable to use the bot in a way that is natural to them.

This is a burden to bot developers as well, who must anticipate and build bindings for each new use case. This is a time consuming and expensive exercise, and despite our best efforts, often results in a mismatch between author expectations and user intent.

On the other end of the spectrum are general purpose scripting and metaprogramming languages that allow users to define their own commands and build embedded domain specific languages. These systems are more flexible, but require a high upfront investment from the user, who must design a language just to define their own commands, and then learn the language itself whilst doing their daily job.

However, there is a fecund middle ground between these two extremes, where users can quickly dictate their own commands and phrases without requiring a Turing Complete language. For example, the user can say "whenever I say *open sesame*, open the settings menu" and the system will learn this command and it will open settings when the user says "open sesame". Or "whenever I say *redo thrice*, repeat the last action three times". Or call a function in a scripting language, open a file, or anything else that the user can think of.

We specially target IDEs, whose users are likely to have some programming fluency, and thus are comfortable configuring their own languages unlike a general audience who expect a pure natural language interface and are unfamiliar with programming. This is the design space which Idiolect occupies.

We describe Idiolect, a programmable system that allows for the creation and use of voice commands. Idiolect is a reconfigurable system that allows users to create their own commands and actions on the fly, by verbally explaining the desired behavior of the system. This imposes some natural constraints, because the user must be able to express their intent in a way that is natural and readily communicated. Anything that requires more complex instructions can be written into a function, and then invoked on the fly with a keyword.

Primarily, Idiolect observes the following design principles: (1) be natural to use, (2) be easy to configure, (3) get out of the user's way as quickly as possible. We believe that these principles are important for a system that is intended to be used by developers, who are more than capable of configuring the system themselves. We also support motor-impaired users who have difficulty typing, or prefer to use a voice interface.

II. Prior Work

Prior work in this area has explored...

III. Action Binding

IntelliJ Platform has over 10^3 possible actions. These actions are bound to keyboard shortcuts, menu items, and toolbar buttons. The user can also bind actions to voice commands. However, the user must first know the name of the action, and then bind it to a voice command. The default grammar was manually curated from this list, using the name to generate a description that is suitable for voice recognition.

IV. Command Prioritization

Highest priority commands are those that enable and disable speech recognition.

Then, user-defined commands.

Then the default commands from a plugin-wide grammar.

The recognizer of last resort are ChatGPT commands. We can use a prompt "What action is the most likely for the phrase "..." out of these actions: ..." and it will select top action as the command.

V. Speech Models

Idiolect integrates with Vosk, a state-of-the-art deep speech system with models for various languages. VoskAPI[2] is open source system that can be used a Java library, which we use.

[1]https://github.com/OpenASR/idiolect

[2]https://github.com/alphacep/vosk-api

For TTS, we use the built-in voices from the parent operating system, via the jAdapterForNativeTTS[3] library.

VI. Internationalizaion and Localization

References

[1] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] K. Elissa, “Title of paper if known,” unpublished.

[5] R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[3]https://github.com/jonelo/jAdapterForNativeTTS