Senior Project Proposal: Vocal Programming Language

Alex MacLean

January 6, 2022

1 Background and Related Work

People have been coding by voice for at least 10 years now. Current systems, such as VoiceCode, Talon, Caster, and Aenea [1] add a layer between an existing voice to text system like Dragon and an existing code editor like Vim, Visual Studio, or DrRacket. This layer looks for command words like "kebab", which causes following words to be formatted kebab-style, or made up words like "lape" for a pair of parentheses (Maybe something like "huh" for the '?' symbol). There are also words or phrases that trigger navigation hotkeys. A system like this has the advantage of working reasonably well for any language, though as with an IDE special commands might be added to allow quickly adding boiler-plate.

One of the major challenges of existing systems seems to be that there is a very steep and long learning curve. One speaker said he had over 2,000 voice commands, and while its likely a user could still be productive with far fewer, even hundreds is a lot to learn. As evidence of the limits of existing technology, they seems to be designed for and used almost exclusively by people who have lost the ability to use a keyboard. While it's great that these people are being served, its telling that anyone who has a choice is sticking with a keyboard instead of spending months to learn a voice to code system. Another limitation seems to be that existing systems often mishear their users, especially when they are not in a silent space with an expensive microphone.

2 Overall Goals

For this project I'm interested in addressing the voice code problem in the speech to text system and the programming language as opposed to just adding a layer between them. Speech to text systems could be specialized for programming so that they resolve noisy input in a way that makes sense given the context of the program, as opposed to natural English language. For example, in a normal context the utterance "wī" would likely mean "why", however in programming "y" seems like a more appropriate choice (especially if that id is in scope or it is preceded by "x"). Existing systems get around this by using a phonetic alphabet, so a command word like "yap" means "y". If the model used by speech to text system were modified, it might be possible to use the normal alphabet and reduce the barrier to entry. A better model would also reduce the error rate, especially in more noisy environments.

On the language design side of the problem, I'm interested in writing a language with voice in mind that can then be compiled into another high level language. Developing this language would require augmenting normal compilation with natural language processing and program synthesis. The surface syntaxes of existing languages are designed to be easy to type and looks good on a screen, but those are not the same concerns when designing a language for voice. A language for vocal programming might have new forms to specify programs in a less line-by-line manner, allow developers some leeway with regard to specific wording, or use a much less symbol heavy surface syntax.

In short, I want to build a system that can be told something like "define a function factorial that takes an int n if n equals zero return one otherwise return n times factorial of n minus one." and produces a program in a language like Racket, JavaScript, or Python.

3 Tentative Sprint Schedule

3.1 Winter Quarter

1. Write Project Proposal and find and read 5 papers related to voice programming, language design, and program synthesis from a description.
2. Find an open source voice to text library, tweak the language model in some way, and build and run the code.
3. Design and train a language model for writing code in specific.
4. Integrate the new model into the open source library and get it running.
5. Gather accuracy metrics for the new and original systems, experiment with tweaking the model to improve accuracy.

3.2 Spring Quarter

1. Write an EBNF and a parser for a simple version of the voice programming language.
2. Get a compiler working from the voice language to another high level language like python, racket, or Java.
3. Integrate the speech to text system and the compiler into some program that writes the code to the screen as it is spoken.
4. Experiment with using large natural language models and macros in the language to make vocal programming better.
5. Upload a cleaned-up version of the code to GitHub, make a simple website with an easy-to-download version of the system. Write the final report.

References

[1] A. Nowogrodzki, "Speaking in code: how to program by voice," *Nature*, vol. 559, pp. 141–142, 2018.

[2] T. Rudd, "Using python to code by voice." https://www.youtube.com/watch?v=8SkdfdXWYaI, Mar. 2013.

[3] E. Shea, "Voice driven development: Who needs a keyboard anyway?." https://www.youtube.com/watch?v=YKuRkGkf5HU, Sept. 2019.