JustSayIt.jl: A Fresh Approach to Open Source Voice Assistant Development

Samuel Omlin[1]

[1]Swiss National Supercomputing Centre (CSCS), ETH Zurich, Lugano, Switzerland

ABSTRACT

We present `JustSayIt.jl`, a software and high-level API for offline, low latency and secure translation of human speech to computer commands or text, leveraging the Vosk Speech Recognition Toolkit. The API includes an unprecedented, highly generic extension to the Julia programming language, which allows to declare arguments in standard function definitions to be obtainable by voice. As a result, it empowers any programmer to quickly write new commands that take arguments from human voice.

Keywords

Julia, Python, Voice Assistant, Speech-to-command, Speech-to-text, Low Latency, Language extension, Voice function arguments

1. Introduction

Leading software companies have heavily invested in voice assistant software since the dawn of the century. However, they naturally prioritize use cases that directly or indirectly bring economic profit. As a result, their developments cover, e.g., the needs of the entertainment sector abundantly, but those of academia and software development only poorly. There is particularly little support for Linux, whereas it is the preferred operating system for many software developers and computational scientists. The open source voice assistant project MyCroft fully supports Linux, but provides little tools that appear helpful for productive work in academia and software development; moreover, adding new skills to MyCroft seems to be complex for average users and appears to require considerable knowledge about the specificities of MyCroft. `JustSayIt.jl` addresses these shortcomings by providing a lightweight framework for easily extensible, offline, low latency, highly accurate and secure speech to command or text translation on Linux, MacOS and Windows.

2. JustSayIt API and software

JustSayIt's high-level API allows to declare arguments in standard Julia function definitions to be obtainable by voice, which constitutes an unprecedented, highly generic extension to the Julia programming language. For such functions, JustSayIt automatically generates a wrapper method that takes care of the complexity of retrieving the arguments from the speakers voice, including interpretation and conversion of the voice arguments to potentially any data type. JustSayIt commands are implemented with such voice argument functions and are triggered by a user definable mapping of command names to functions. As a result, it empowers programmers without knowledge of speech recognition to quickly write new commands that take their arguments from the speakers voice.

```julia
1  using JustSayIt
2  using JustSayIt.API
3  using DefaultApplication
4  @enum Day today tomorrow
5
6  #1) Define a custom weather forecast search function
7  @voiceargs day=>(valid_input_auto=true) function
8  weather(day::Day)
9      DefaultApplication.open(
10       "https://www.google.com/search?q=weather+$day")
11 end
12
13 #2) Define command name to function mapping,
14 #   calling the custom function.
15 commands = Dict("help"    => Help.help,
16                 "weather" => weather)
17
18 # 3) Start JustSayIt with the custom commands.
19 start(commands=commands)
```

Fig. 1. Definition and usage of custom weather forecast search function in JustSayIt.

Fig. 1 shows an executable example of 1) a simple voice argument function, which enables a quick weather forecast search by voice (lines 7-11), 2) a definition of a command name to function mapping (lines 14-15), which uses the function defined above, and 3) the launching of the JustSayIt software with the defined command name to function mapping (line 18).
JustSayIt provides a lot of powerful functionality that does not require programming (Fig. 2); it enables mapping of command names

—to predefined functions as, e.g., for typing text with the keyboard (line 7) or controlling the mouse (lines 8-12),

—to keyboard shortcuts (lines 13-16), and

—to sequences of functions or keyboard shortcuts (lines 17-20).

The predefined function for typing text (Fig. 2, line 7) enables the dictation of full text, words, letters or digits. Each of the modes supports a set of keywords which can trigger some immediate action or modify the handling of subsequent speech input. A particularity of JustSayIt's speech to text approach is its strategy for the recognition of these keywords: the speech is analyzed in word groups which are naturally delimited by longer silences; then, keywords are only considered as such if their word group does not contain anything else then keywords. This allows to determine whether a word that is recognized as a keyword should trigger some action or be typed instead.

```julia
1  using JustSayIt
2
3  #1) Define mapping of command names to functions,
4  #   keyboard shortcuts and command sequences.
5  commands = Dict(
6      "help"      => Help.help,
7      "type"      => Keyboard.type,
8      "ma"        => Mouse.click_left,
9      "middle"    => Mouse.click_middle,
10     "right"     => Mouse.click_right,
11     "hold"      => Mouse.press_left,
12     "release"   => Mouse.release_left,
13     "undo"      => (Key.ctrl, 'z'),
14     "redo"      => (Key.ctrl, Key.shift, 'z'),
15     "page up"   => Key.page_up,
16     "page down" => Key.page_down,
17     "take"      => [Mouse.click_double,
18                     (Key.ctrl, 'c')],
19     "replace"   => [Mouse.click_double,
20                     (Key.ctrl, 'v')]
21     )
22
23 #2) Start JustSayIt, activating max speed
24 #   recognition for a subset of the commands.
25 start(commands=commands,
26       type_languages=["en-us", "fr"],
27       max_speed_subset=["ma", "middle", "right",
28       "hold", "release", "page up", "page down",
29       "take"])
```

Fig. 2. Usage of some of JustSayIt's powerful functionalities that do not require programming: making use of predefined functions, keyboard shortcuts and command sequences.

JustSayIt is designed for full multi-language speech-to-command and speech-to-text support for the over twenty languages available with the Vosk Speech Recognition Toolkit[1] [7]. Languages are selectable with keyword arguments (e.g., Fig. 2, line 26).

3. Speech recognition algorithm

JustSayIt implements a novel algorithm for high performance context dependent recognition of spoken commands which leverages the Vosk Speech Recognition Toolkit [7] (which in turn relies on the Kaldi speech recognition toolkit [6]). A specialized high performance recognizer is defined for each function argument that is obtainable by voice and has a restriction on the valid input. In addition, when beneficial for recognition accuracy, the recognizer for a voice argument is generated dynamically depending on the command path taken before the argument. To enable minimal latency for single word commands (latency refers here to the time elapsed between a command is spoken and executed), the latter can be triggered when appropriate upon bare recognition of the corresponding sounds without waiting for silence as normally done for the confirmation of recognitions (this behaviour can be activated for each command individually, see Fig. 2, lines 26-28). Thus, JustSayIt is suitable for commands where a perceivable latency would be unacceptable, as, e.g., mouse clicks. For example, for the single word command "ma" (mapped to the left mouse button) latencies between 5 ms and 24 ms with a median of 6 ms were measured on a regular notebook[2]. JustSayIt achieves this low latency using only one CPU core and can therefore run continuously without harming the computer usage experience.

[1]supported as of today: English, French, German (partial) and Spanish (partial)

[2]100 samples were taken on a Lenovo ThinkPad P15 Gen 1 notebook with a 12-core Intel i7-10750H CPU @ 2.60GHz and 64 GB DDR4 memory.

4. Python integration

JustSayIt makes best use of both Julia and Python rather than limiting itself to one language: it leverages Julia's performance and metaprogramming capabilities[3] and Python's larger ecosystem where no Julia package is considered suitable. The Vosk Speech Recognition Toolkit, a C++ library, is used via its convenient Python bindings, available as the Python module `vosk`. JustSayIt relies on `PyCall.jl` [4] and `Conda.jl` [1] for a straightforward integration of Python modules as, e.g., `vosk`, `pynput` [5] and `sounddevice` [2]. All Python dependencies are fully automatically installed into an isolated Python environment.

5. Conclusions

The JustSayIt API's possibility to declare arguments in standard Julia function definitions to be obtainable by voice constitutes an unprecedented, highly generic extension to the Julia programming language. Moreover, it allows JustSayIt to combine accuracy, performance and security with extensibility of particular ease and generality. The great and effortless extensibility and the possibility for straightforward integration of Python modules into JustSayIt provide an ideal basis to unite the world-wide Julia and Python communities in the development of this open source project. The JustSayIt project demonstrates that the development of our future voice assistants can take a fresh and new path that is neither driven by the priorities and economic interests of global software companies nor by a small open source community of speech recognition experts; instead, the entire world-wide open source community is empowered to contribute in shaping our future daily assistants.

6. References

[1] Guillaume Fraux. Conda.jl: Conda managing Julia binary dependencies. `https://github.com/JuliaPy/Conda.jl`, 2015.

[2] Matthias Geier. sounddevice: Play and Record Sound with Python. `https://github.com/spatialaudio/python-sounddevice`, 2015.

[3] Mike Innes, Julia Computing, and contributors. MacroTools.jl: a library of tools for working with Julia code and expressions. `https://github.com/FluxML/MacroTools.jl`, 2015.

[4] Steven G Johnson and contributors. PyCall.jl: Package to call Python functions from the Julia language. `https://github.com/JuliaPy/PyCall.jl`, 2022.

[5] Moses Palmer. pynput. `https://github.com/spatialaudio/python-sounddevice`, 2015.

[6] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*. IEEE Signal Processing Society, 2011. IEEE Catalog No.: CFP11SRW-USB.

[7] Nickolay V. Shmyrev and other contributors. Vosk Speech Recognition Toolkit: Offline speech recognition API for Android, iOS, Raspberry Pi and servers with Python, Java, C# and Node. `https://github.com/alphacep/vosk-api`, 2020.

[3]using `MacroTools.jl` [3] where helpful