CODE BY VOICE USING SILVIUS

Abhishek Chand[1], Subhojeet Chakraborty[2], Abhinava Anand[3], Swapnil Dhande[4], Mansa Mane[5], P.S. Kulkarni[4]

[1]Student, Department of Computer Engineering, NBN Sinhgad School of Engineering, Maharashtra, India
[2] Student, Department of Computer Engineering, NBN Sinhgad School of Engineering, Maharashtra, India
[3] Student, Department of Computer Engineering, NBN Sinhgad School of Engineering, Maharashtra, India
[4] Student, Department of Computer Engineering, NBN Sinhgad School of Engineering, Maharashtra, India
[5] Assistant Professor, Department of Computer Engineering, NBN Sinhgad School of Engineering, Maharashtra, India
[6] Assistant Professor, Department of Computer Engineering, NBN Sinhgad School of Engineering, Maharashtra, India

ABSTRACT

Carpal tunnel and repetitive strain injuries can prevent programmers from typing for months at a time. Fortunately, it is possible to replace the keyboard with speech recognition. The key is to develop a voice grammar customized for programming. A community has evolved around hacking the commercial Dragon NaturallySpeaking to use custom grammars, but this method suffers from fragmentation, a steep learning curve, and frustrating installation difficulties. In an attempt to make voice coding more accessible, David created a new speech recognition system called Silvius, built on open-source software with free speech models. It can run on cloud servers for ease of setup, or locally for the best latency. The hope is that Silvius will lower the bar for experimentation and innovation, and encourage ordinary programmers to try voice coding, instead of waiting until a crippling injury throws them in at the deep end.

Keyword: - *Speech Recognition, Deep Learning, Neural Networks, Recurrent Neural Networks*

---

1. Introduction

Silvius is an open source system for writing code by voice. Speak characters and words and they are typed for you automatically. This helps in writing code without keyboard and thus can help physically disabled people. It is the offspring of Dragon NaturallySpeaking and Aenea. Silvius works by piping the microphone output to a server, and the server responds with the sentences it recognize. The parsed speech is then run through a grammar that produces virtual keyboard strokes. Looking at Silvius, you might think it would be slow as the audio has to take a roundtrip to the server—but it actually seems surprisingly snappy.
I think the strong innovation in Silvius is the fact that it relies on a platform-agnostic speech recognition algorithm—in the end that might allow for something that will work across all platforms.

2. Literature Review

2.1 Evolution of Voice Coding

In 1991, NatLink was created by Joel Gould of Dragon Systems to allow Python macros. It was a compatibility module (like NatText) which allowed you to write NatSpeak command macros in Python. It worked with all versions of NatSpeak. It was free and open-source, freely distributable.

In 2008, Christo Butcher writes Dragonfly, providing a framework for Python grammars. It is a Python package, which offers a high-level object model and allows its users to easily write scripts, macros, and programs, which use speech recognition.

In 2013, Tavis Rudd gives talk at PyCon about custom voice coding on Linux.

In 2014, Aenea by Alex Roper recreates full Linux voice coding support. It is a system to allow speech recognition via Dragonfly on one computer to send events to another.

3. KALDI

Kaldi is a toolkit for speech recognition written in C++ and licensed under the Apache License v2.0. Kaldi is intended for use by speech recognition researchers. Kaldi is similar in aims and scope to Hidden Markov Model. The goal is to have modern and flexible code, written in C++, which is easy to modify and extend. Important features include:

- Code-level integration with Finite State Transducers (FSTs)
- Extensive linear algebra support
- Extensible design
- Open license
- Complete recipes

3.1 Deep Neural Networks in Kaldi

An active area of research like this is difficult for a toolkit like Kaldi to support well, because the state of the art changes constantly which means code changes are required to keep up, and architectural decisions may need to be rethought.

There are currently three separate codebases for deep neural nets in Kaldi. All are still active in the sense that the up-to-date recipes refer to all of them. The first one "nnet1" is located in code subdirectories nnet/ and nnetbin/, and is primiarly maintained by Karel Vesely. The second is located in code subdirectories nnet2/ and nnet2bin/, and is primarily maintained by Daniel Povey (this code was originally based on an earlier version of Karel's code, but it has been extensively rewritten). The third is located in code subdirectories nnet3/ and nnet3bin/, and Dan's previous work on nnet2 will shift to the nnet3 setup.

3.2 Online decoding in Kaldi

By "online decoding" we mean decoding where the features are coming in in real time, and you don't want to wait until all the audio is captured before starting the online decoding.

The approach that had been taken with Kaldi was to focus for the first few years on off-line recognition, in order to reach state of the art performance as quickly as possible. Now more of an effort is made to support online decoding.

There are two online-decoding setups: the "old" online-decoding setup, in the subdirectories online/ and onlinebin/, and the "new" decoding setup, in online2/ and online2bin/. The "old" online-decoding setup is now deprecated, and may eventually be removed from the trunk (but remain in ^/branches/complete).

3.3 Keyword Search in Kaldi

Keyword search module in Kaldi uses the following features:

- Lattice indexing for fast keyword retrieval.

- Proxy keywords to handle out-of-vocabulary (OOV) problem.

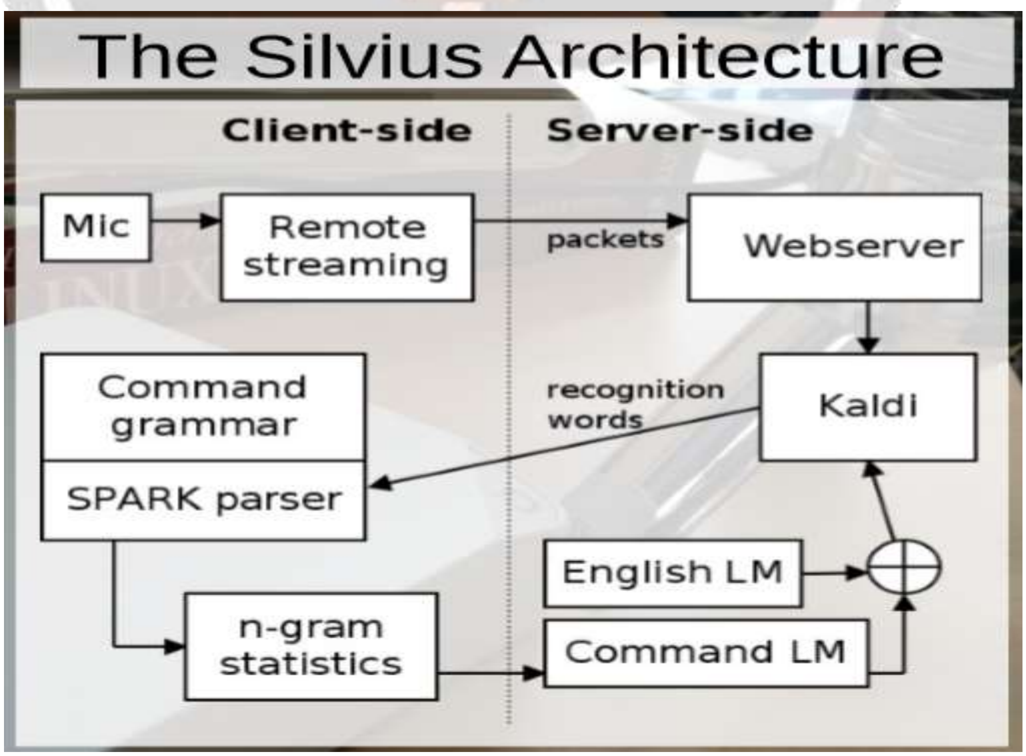
4. BASIC VOICE GRAMMAR DESIGN

- NATO-esque alphabet
  - arch, bravo, char, delta, echo, fox, golf, hotel, …
- Symbols and characters
  - 0-9, space, “slap” for enter, “act” for escape, ...
  - ( ) [ ] < > { } are “l”/”r” + “en”/”ack”/”angle”/”ace”
- English words
  - sentence hello there → Hello there
  - score merge sort → merge_sort
- Chaining: say sequences without pausing

5. SILVIUS

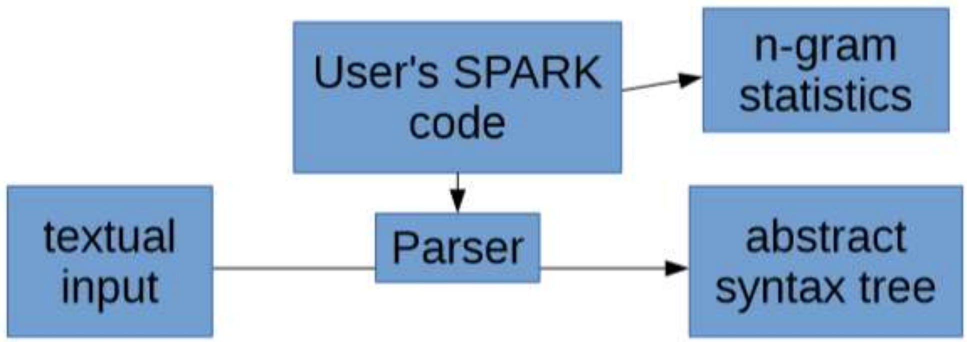
Silvius is an open source system for writing code by voice. Speak characters, words and they are typed for you automatically. Silvius works by piping the microphone output to a server, and the server responds with the sentences it recognize. The parsed speech is then run through a grammar that produces virtual keyboard strokes.

5.1 Architecture



5.2 Silvius Grammars:

It is written in Python with SPARK parsing toolkit. It creates a parser tree with meta-Python objects. It can walk the parser tree to generate n-gram LM. The parser then converts text to an abstract syntax tree. The AST is walked and commands are executed. It works like a compiler-compiler with introspection.



6. ADVANTAGES

- It can run locally or in cloud
- It can work on user-provided custom speech grammar
- low computing resources required
- simple installation requirements
- a true voice keyboard

7. CONCLUSIONS

Thus, we conclude that when we can’t type we can harness speech recognition through Silvius and code by voice. This can help in building new ways of interacting with computers.

8. REFERENCES

[1] Mirco Ravanelli, Philemon Brakel, Maurizio Omologo, Yoshua Bengio: A NETWORK OF DEEP NEURAL NETWORKS FOR DISTANT SPEECH RECOGNITION, 2017

[2] Abdulghani Ali Ahmed and Nurul Amirah Abdullah, “Real Time Detection of Phishing Websites” IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), 2016

[3] X. Chen, A. Ragnil, J. Vasilakes, X. Liu, K. Knilll, MJ.F. Gales: Recurrent neural network language models for keyword search. 2017

[4] Sri Harsha Dumpala, Sunil Kumar Kopparapu : Improved Speaker Recognition System for Stressed Speech using Deep Neural Networks, 2017

[5] Zhiyuan Tang, Dong Wang, Zhiyong Zhang: Recurrent neural network training with dark knowledge transfer, 2016

[6] G.SaonM.Picheny: Recent advances in conversational speech recognition using convolutional and recurrent neural networks, 2017

[7] Ebru Arisoy, Abhinav Sethy, Bhuvana Ramabhadran, Stanley Chen: BIDIRECTIONAL RECURRENT NEURAL NETWORK LANGUAGE MODELS FOR AUTOMATIC SPEECH RECOGNITION, 2015

[8] http://elfery.net/projects/silvius.html