2ZB-06

音声認識を用いたソフトウェア開発環境の構成と実装

Organization of Development Environment Using Voice Recognition

谷口 英貴

東京電機大学大学院

1, はじめに

近年のハードウェアの進歩は目を見張るものがありタッチパネルを始めとする、より簡単な入力処理を可能とするデバイスが次々と発明されている。しかしこれらのデバイスは直観的な操作を可能とする反面、ソフトウェア開発における改善は見受けられず、タッチパネルなどによるコーディングは困難を極めているというのが現状である。本報告では音声認識エンジン Julius を用いた、より柔軟かつ最新デバイスに適応可能なソフトウェア開発環境の構築を目指す。音声認識を用いたソフトウェア開発環境の構築はより柔軟かつ手軽なプログラミングを可能とし、タブレット PC やスマートフォンにおけるスムーズなコーディングを実現すると考えられる。また、少量の音声情報のみから目的の処理を見つけることのできる、プログラミング知識の乏しい人であってもソフトウェアの開発が可能となるようなモデルの提案を行う。

2, 音声認識エンジン Julius

Julius はオープンソースの音声認識エンジンであり、主に日本語を扱った音声認識モデルにおいて幅広く利用されている。また Julius は独自に単語辞書を持つことができ、単語とその読みの定義や、単語毎の生起確率を指定することが可能であり、音声認識システム毎の目的に合わせた調整及び認識の最適化を行うことが可能である。音声認識を用いたソフトウェア開発環境の構築においては言語毎の予約語や標準キーワードの登録により誤検出を防ぐことができる。

3, 先行研究

2013 年に Ruddtwo による発表[1]において、音声情報のみを用いて Python のコーディングをする手法が提案された。

Organization of Development Environment Using Voice Recognition
Graduate School of Tokyo Denki University

Ruddtwo の報告ではエディタ操作に重きが置かれ、マイクを通し肉声情報を入力情報として扱うという点を除き、これまでの手指を用いたコーディングと根本的な違いは無い。また Chirstine により、Java と音声認識ソフト Dragon を用いた音声認識による Java プログラミング手法が提案された[2]。Chirstine の報告では複数の音声認識候補に対しひとつの処理を当てはめる手法を取っており、例えば「do」,「do the following」, 「loop body」, 「body」,「for body」,s「while body」といった音声入力は全て単一のループ処理に対応される。

4, 提案手法

これ等の先行研究を踏まえ、本報告では肉声情報のみから目的とするソースコードを自動生成するという、既存の手法とは異なるプログラミング手法の提案を行う。先行研究では利用者がソフトウェアの仕様を考慮して音声入力を行わねばならないが、本研究ではあらゆる音声入力に対してもプログラム側が何かしらの関連付けを行い、出力結果を生成する。

本プログラムは取得された音声情報を文章化し、さらにその文章に含まれる単語を「変数」「文字列」「関数」「クラス」のいずれかとして扱う。例として「hoge イコール 1 足す 5」,「FizzBuzz 100」、「情報処理」という音声入力が存在する場合、これらは順番に、hoge という変数に 1+5 の結果で初期化し宣言する処理、1 から 100 までの FizzBuzz の結果の表示をする関数の呼び出す処理、”情報処理”という文字列を出力する処理として扱われる。

今回の報告では一般的なパソコンと比べキーボードからの入力にラグのある Raspberry Pi などの廉価ボードにおける、既存の入力デバイスからの代替えとなるようなシステムの構築の検証を行った。またこの試みでは、比較

的構文の規則が易しく、また機能が多用である Python を用いた。

5, 動作の流れ

プログラムの一連の動作を示す。

(1), 音声認識プログラム Julius を起動し、ユーザからの音声入力を受け取る。この際に独自の単語辞書を設定することで処理に適した設定を行う。

(2), 取得した音声情報から有効な文章を切り出す。Julius は音声からテキストへと変換をする際に音韻毎に句読点などが混ざることがあるため、これらを取り除く。

(3),編集したテキストの構文解析を行う。(2)の結果を基に各単語が適応する関数・クラスが存在するかを調べ、存在する場合はそれらに割り当てる。適応する関数・クラスが存在せず、単語が’=’を含む場合は単語i, ‘=’, 単語jとして処理し変数の宣言と初期化を行う。いずれにも該当しない場合は全て文字列の出力処理として扱う。

(4),(3)で得られたソースコードを保存・実行する。

この一連の動作をプログラム側が自動で行うことにより、プログラマはマイクに向かって話しかけるのみでソースコードの生成及びデバッグを行うことが可能である。

6, まとめ

現在普及しているタブレット PC やスマートフォンは、直観的な操作をユーザに提供しつつも、プログラミング環境における改善は見受けられない。そこで本研究では音声認識エンジン Julius を用いたソフトウェア開発環境の構築を行い、既存の手法に比べより柔軟な入力処理を行うことのできるソフトウェア開発環境の構成を行った。しかし、現時点では音声入力の正解率が低く予想とは異なる出力結果となってしまうことが多々あり、また登録がされていない処理は全て文字列として扱われてしまうため、実用性に乏しいと言わざるを得ない。今後の発展として Ruddtwo が提案した音声によるソースエディタ操作を本プログラムへの適応を考える。現状では本プログラムは誤ったソースコードを出力した場合、その修正はプログラマが手直しで直さねばならず、既存のプログミング手法の域を出ないためである。また、今の時点では GUI プログラミングに対応をしていないため、Numpy や Pygame などへの適応を考える。声のみによるエディタ操作を採用することによる、手作業から完全に脱却したプログラミング環境の構築をこのテーマのゴールとし、その実装を目指す。

参考文献

1) Tavis Ruddtwo “Using Python to Code by Voice”, Pycon 2013.
2) Christine Matsuoka “Java Programming Using Voice Input: adding Java Support to Voice Code”.
3) Mubashir Ayub and Muhammad Asjad Saleem “A Speech Recognition based Approach for Development in C++”, IJCSNS International Journal of Computer Science and Network Security, Vol.12 No.10, Oct 2012.
4) Andrew Bagel, Susan L. Graham “An Assessment of a Speech-Based Programming Environment”, 2006 IEEE Symposium on Visual Languages and Human-Centric Computing, pages 116-120.
5) Steven Bird, Ewan Klein, Edward Loper “Natural Language Processing with Python”, O’Reilly Media, 2009.
6) Wes Mckinney “Python for Data Analysis”, O’Reilly Media, 2012
7) Alain Désilets, David C. Fox, Stuart Norton, “Voice Code: an Innovative Speech Interface for Programming-by-Voice”, CHI 2006, April 22-27.