



Vocal Chess

上杉真由
CHOI HYUNSEOK





目次

- + 目的
- + 概要
- + 工夫した点
- + デモ
- + 展望

目的

課題

- ・頭の中で先の手を読むのは難しい
- ・いろんな体勢でチェスがしたい

→仮プレイモードを兼ね備えた音声対話式チェスを作ろう！



実装に用いたもの

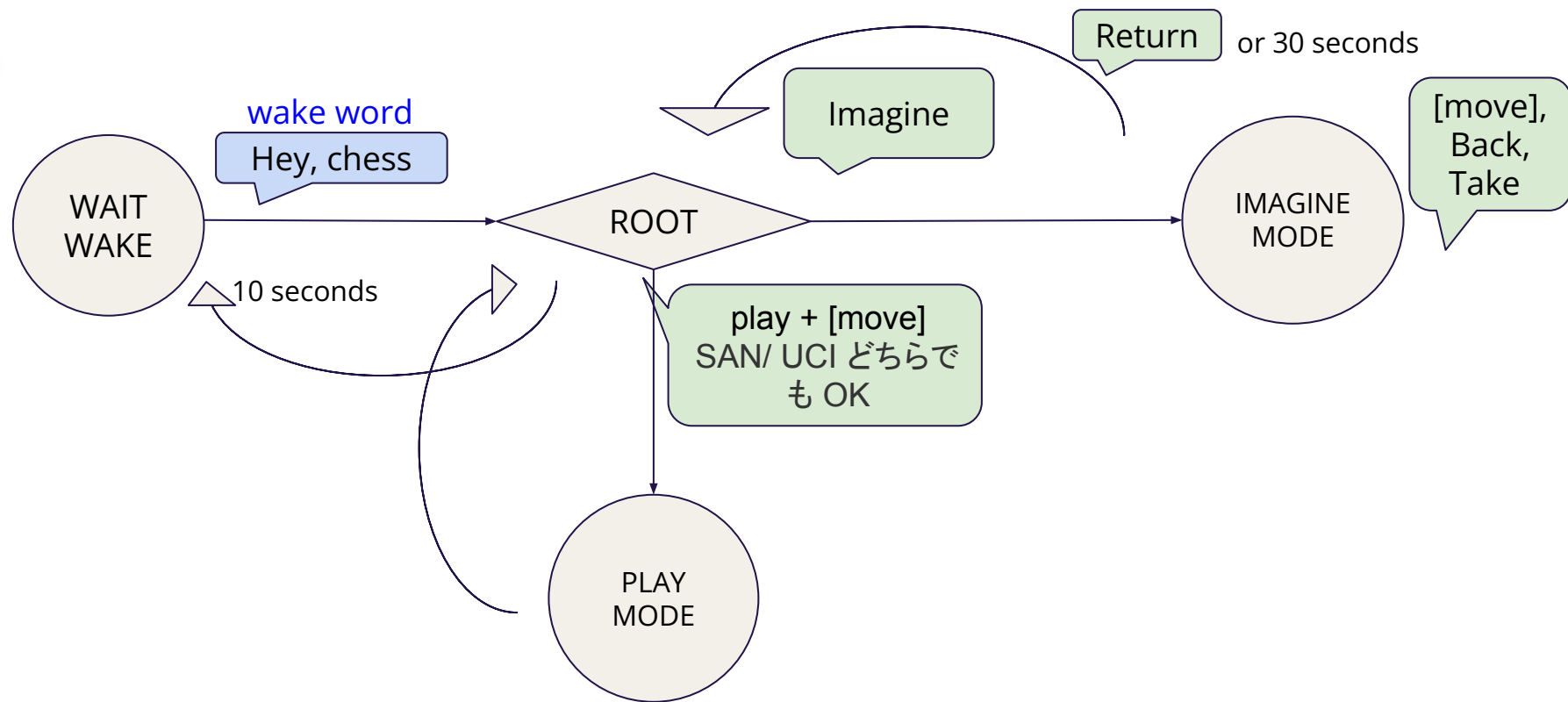
GUI : tkinter

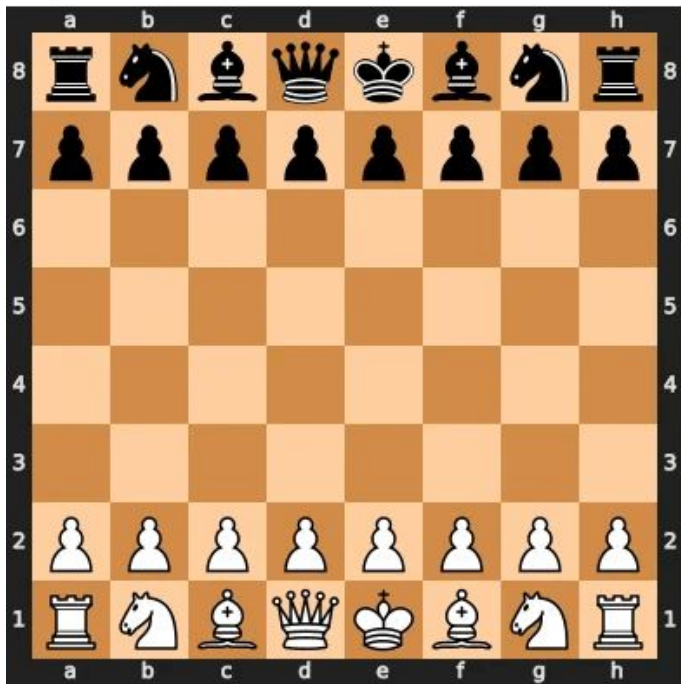
Chess Engine: stockfish

Voice recognition: VOSK(vosk-model-small-en-us-0.15)

Wake word detection: local-wake

Vocal Chessのフローチャート :FSM





Move表現方法

SAN (Standard Algebraic Notation)

e.g. e4 (pawnをe4に移動)

Nf3 (kNightをf3に移動)

UCI (Universal Chess Interface)

e.g. e2e4 (e2をe3に移動)

g1f3 (g1をf3に移動)

詳細

ROOT

自分が動かし終わると相手の手が自動で打たれる。

10秒間無入力でWAITWAKEへ

音声コマンド:

“Play [move]”

“Imagine” : IMAGINE MODEに入る

IMAGINE MODE

自分の駒と相手の駒をそれぞれ交互に動かせる ⇒ 思考の視覚化

30秒間無入力でROOTへ

音声コマンド:

“[move]”

“BACK” : 1手前の盤面に戻る

“RETURN” : 元の盤面に戻る (Imagine 終了)

”TAKE” : Engine moveの実行

実装上の工夫: wake word

Wake word 検出のキーポイント

- 話者依存性が少ない
- 誤検出が少ない

実装上の工夫: wake word



古典的手法 (MFCC + DTW)

メリット

実装が単純

デメリット

話者依存性がありノイズに弱い

深層学習ベース (openWakeWordなど)

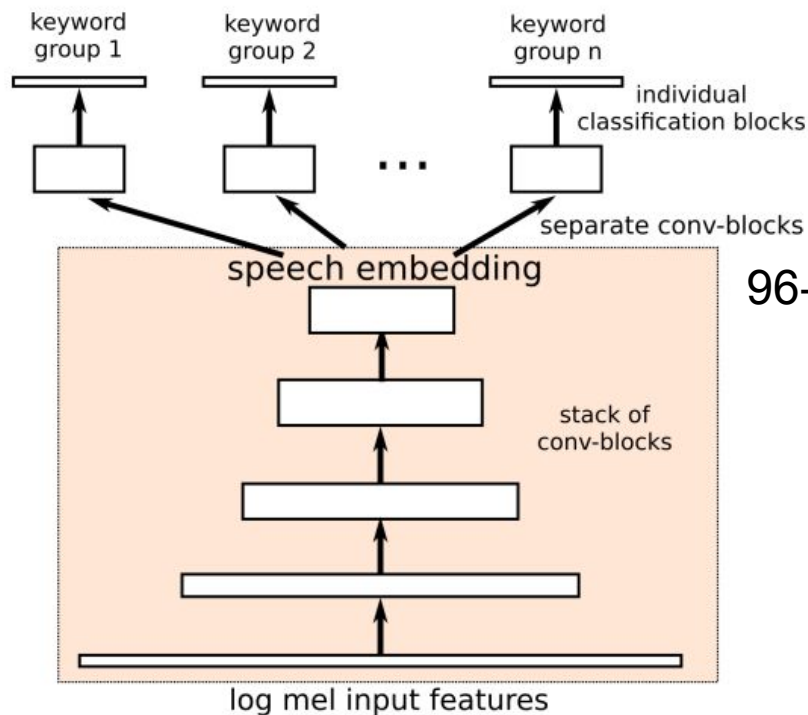
メリット

話者依存性が少なくノイズに強い

デメリット

wake word ごとの学習が必要

実装上の工夫: wake word



96-dim vector (80-ms frame)

<https://arxiv.org/pdf/2002.01322>

実装上の工夫：自然発話を考慮したコマンド抽出

VoskのKaldi_rocognizer

Grammarを設定できる

部分的な結果と最終結果がjson形式で出力される

`{'partial': 'one zero zero zero one'}` : 発話中に逐次返ってくる部分的な認識結果

`{"text" : "one zero zero zero one"}` : 発話が一区切りしたときに返ってくる最終結果

実装上の工夫：自然発話を考慮したコマンド抽出

雑談を含む会話から有効なコマンドを抽出したい

“play pawn to e4” ⇒ [“PLAY”, “PAWN”, “TWO”, “E”, “FOUR”] ⇒ e4

“I don’t know, maybe pawn e4?” ⇒ [..., “PAWN”, “E”, “FOUR”] ⇒ e4

“a8 rook to a7” ⇒ [“A”, “EIGHT”, “ROOK”, “TWO”, “A”, “SEVEN”] ⇒ a8a7

実装上の工夫：自然発話を考慮したコマンド抽出

単一コマンドは単独で認識されたときのみ処理

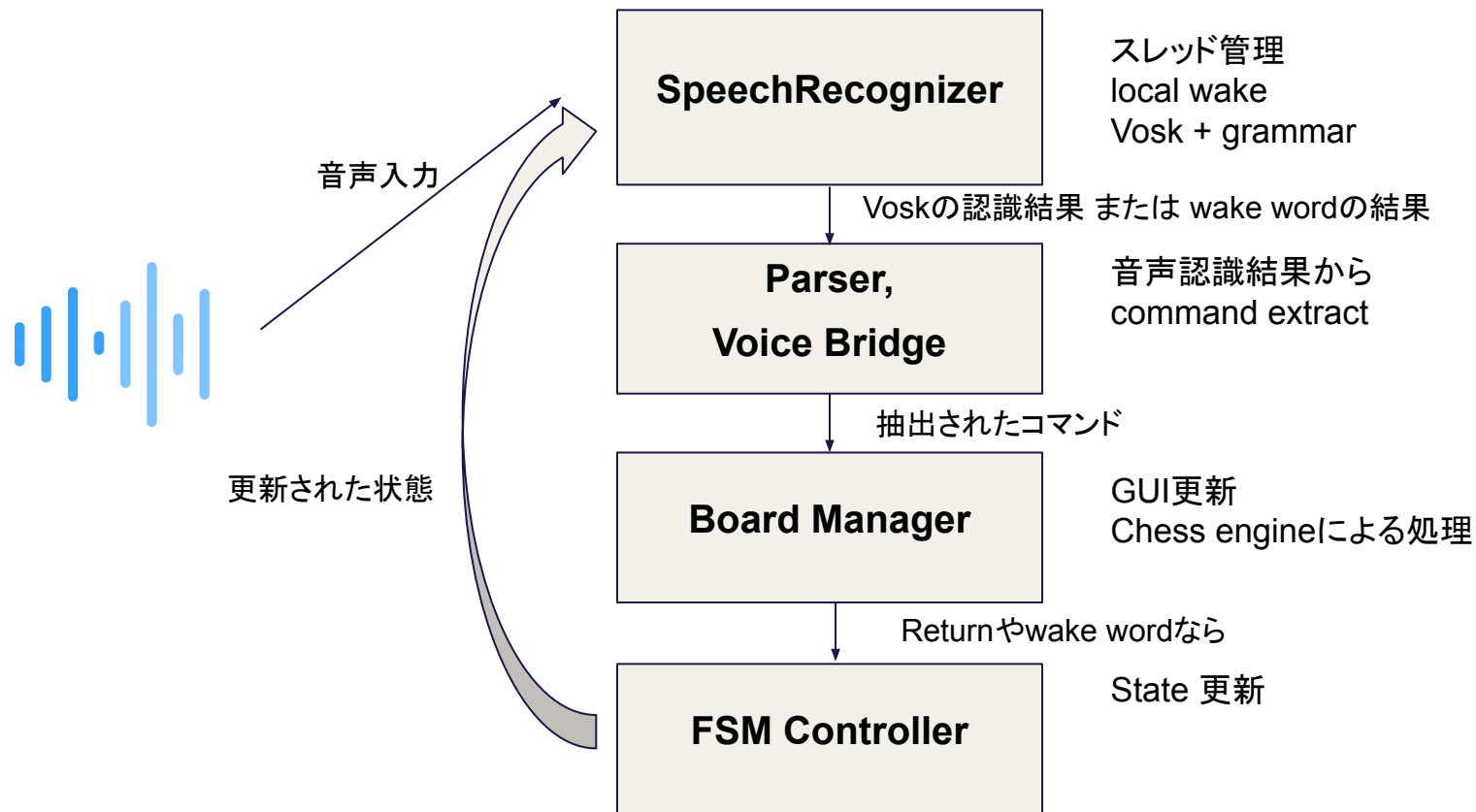
e.g. {"text" : "return"}

Move コマンドは

1. トークン数の上限によるフィルタリング (≤ 10)
2. トークン列からpiece→file→rank(or UCI)順を満たすトークンを抽出

e.g. {"text" : "one", "return", "pawn", "two", "e", "four"}

Vocal Chess入力処理の概要



デモ





今後の展望

多様な表現形式への対応強化

アンパッサン(en passant)などの特殊手を含む、より柔軟で自然な音声入力を受け付けられるように拡張

対話的な機能の追加

説明機能・局面分析・ゲームレビューなどを導入し、ユーザと対話しながら学習できるインタラクティブなシステムに



Thank You!

