Information Technologies for Industrial Engineers

เทคโนโลยีสารสนเทศสำหรับวิศวกรอุตสาหการ

Speech Recognition Application

Speech recognition

- Process of identifying a human voice.
- Takes an audio file of a speaker, recognizes the words in the audio, and converts the words into text.
- Usually, it is combined with natural language processing (NLP).

Steps in speech recognition

- Recording
    - The user's voice is kept as an audio signal after being recorded.
- Sampling
    - Sound wave is converted into discrete samples at a particular frequency.
- Transforming to frequency domain
    - The audio signal's time domain is changed to its frequency domain in this stage.

Steps in speech recognition (cont.)

- Information Extraction from Audio
  - The audio is transformed into a vector format.
- Recognition of extracted information
  - Machine learning / prediction

Source

Models in speech recognition

- Hidden Markov Models (HMMs)
- Deep Neural Networks (DNNs)
- Convolutional Neural Networks (CNNs)
- Transformer-based model

Speech Command Recognizer

- CNN based (source)
- Recognizes of spoken commands comprised of simple isolated English words
  - Ten digits from `zero` to `nine`
  - `up`, `down`, `left`, `right`, `go`, `stop`
  - `yes`, `no`
  - `unknown word`, `background noise`

Source

Setting up

- `npm install @tensorflow-models/speech-commands@0.4.2`
    - The newer version does not work.
- `npm install -D vite-plugin-node-polyfills`
    - Use *polyfill* (code that implements a feature that Vite does not natively support.)

./vite.config.js

```js
import { defineConfig } from "vite";
import react from "@vitejs/plugin-react-swc";
import { nodePolyfills } from "vite-plugin-node-polyfills";

// https://vitejs.dev/config/
export default defineConfig({
  plugins: [
    react(),
    {
      ...nodePolyfills({
        // To exclude specific polyfills, add them to this list.
        exclude: [
          "fs", // Excludes the polyfill for `fs` and `node:fs`.
        ],
        // Whether to polyfill specific globals.
        globals: {
          Buffer: true, // can also be 'build', 'dev', or false
          global: true,
          process: true,
        },
        // Whether to polyfill `node:` protocol imports.
        protocolImports: true,
      }),
      apply: "build",
    },
  ],
  define: {
    "process.env": {},
  },
});
```

./src/model.ts

```
import * as speech from "@tensorflow-models/speech-commands";

export async function load_model() {
  try {
    const recognizer = speech.create("BROWSER_FFT");
    await recognizer.ensureModelLoaded();
    const labels = recognizer.wordLabels();
    return { model: recognizer, labels };
  } catch (err) {
    console.log(err);
    return { model: null, labels: [] as string[] };
  }
}
```

Code

`./src/App.tsx`

https://gist.github.com/nnnpooh/88ae4e520fad0ad12582114cb0edca22#file-app-tsx