SpeechCloud API

2016-05-02

Jan Švec
<honzas@ntis.zcu.cz>

SpeechCloud API

API propojující hlavní řečové technologie

- **Rozpoznávání řeči**
  (Automatic Speech Recognition - ASR)
- **Syntéza řeči**
  (Text-to-Speech - TTS)
- **Porozumění řeči**
  (Semantic Entity Detection - SED)

Navrženo pro cloud

- **klient**
  požaduje služby ASR/TTS/SED
- **frontend**
  vydává konfiguraci pro připojení k SIP ústředně a k workerům
- **worker**
  propojen s audio, provádí ASR, TTS, SED
- **SIP ústředna**
  ústředna propojující audio streamy klient ↔ worker

Předávání informací

Datové zprávy

- JSON zprávy klient ↔ frontend ↔ worker
- WebSocket protokol (+JSON+JSON Schema)
- klient ⇢ worker: **metody**
  začni rozpoznávat, nastav SED, syntetizuj
- klient ⇠ worker: **události**
  rozpoznávám, syntéza dokončena, rozpoznán výsledek

Audio stream

- RTP stream ↔ SIP ústředna ↔ worker
- SIP protokol (+RTP audio)

WebSocket protokol

Full-duplex protokol nad TCP

Iniciován pomocí HTTP, poté Upgrade request

Interface (JavaScript):

- `new WebSocket(`**`URL`**`)` nový WS na URL
- `onopen` při úspěšném navázání WS
- `send(data)` zaslání dat do WS
- `onmessage` při přijetí dat z WS
- `onerror` při chybě
- `onclose` při uzavření WS protistranou
- `close()` uzavření WS

SIP protokol (Session Initiation Protocol)

IP telefonie, UDP (std port 5060), SIP-over-WS

SIP - signalizace

- registrace na ústředně (jméno + heslo)
- navázání spojení
  (protokol SDP - vyjednání RTP parametrů)
- řízení spojení, ukončení

RTP - audio stream

- přenos vlastního audio pomocí kodeků
- IP cesta RTP paketů nemusí být totožná se SIP pakety (přímá komunikace klientů)

Získání konfigurace ze SC

1) SpeechCloud URI

https://cak.zcu.cz:9443/v1/speechcloud/**numbers**
SpeechCloud aplikace - konfigurace konkrétního ASR/TTS/SED engine

2) Získání konfigurace ze SpeechCloud URI

HTTP GET, výsledek JSON:

```
{ "client_wss": WebSocket pro připojení klienta,
  "sip_uri": SIP URI pro registraci,
              (např. "sip:cak.zcu.cz:6060")
  "sip_wss": SIP URI pro SIP-over-WebSocket
    (z browseru, např. "wss://cak.zcu.cz:9445"),
  "client_id": ID klienta (logování),
  "sip_username": SIP username,
  "sip_password": SIP heslo,
...}
```

Připojení na SC

3) Registrace na SIP ústřednu

Přímé použití SIP (knihovna)

SIP-over-WebSocket (web browser)

4) Otevření řídícího WebSocketu

`client_wss` URI pro řídící WS

Až po úspěšné registraci

Otevřený WS znamená vyhrazený worker

Není-li dost workerů, selže otevření WS

5) SIP invite libovolného čísla

SIP ústředna propojí s workerem

Registrace na SIP ústřednu

```
# str() provides conversion from unicode to 8bit strings
acc_cfg = pj.AccountConfig()

# Local client ID (PJSIP requires it as a sip: URI, so prepend the scheme)
acc_cfg.id = 'sip:'+str(json_data['sip_username'])

# Registrar URI
acc_cfg.reg_uri = str(json_data['sip_uri'])

# Call proxy, the same as registrar
acc_cfg.proxy = [ str(json_data['sip_uri']) ]

# Registration credentials
# client_id is virtually the part of sip_username before @
# sip_password is a one-time password used for client registration
acc_cfg.auth_cred = [ pj.AuthCred("*",
            str(json_data['client_id']), str(json_data['sip_password'])) ]
```

Registrace na SIP ústřednu SIP-over-WebSocket

Použití knihovny SIP.js (http://sipjs.com/)

```
uaconfig = {
    // SIP-over-WS URI
    wsServers: config.sip_wss,
    // SIP username
    uri: config.sip_username,
     // SIP password
    password: config.sip_password
};

ua = new SIP.UA(uaconfig);
```

Aktivace session

6) Vyčkat na zprávu sc_start_session

```
{ "type": "sc_start_session",                    // jméno události
  "session_id": "pDKdjJmKaurQGpMK5He8XH",        // session id (logování)
  "schema": {                                    // JSON schéma komunikace
    "events": {                                  // seznam událostí
      "asr_signal": { },
      "asr_recognizing": { },   ...
    },
    "methods": {                                 // seznam metod
      "tts_synthesize": { },
      "asr_recognize": { },  ...
    },
    "description": "SpeechCloud schema of events and methods",
    "type": "object",
    "oneOf": [{"$ref": "#\/events\/asr_signal"}, ... ]
  }
}
```

Průběh a ukončení session

7) Událost asr_ready

vygenerované po spuštění session a obdržení prvního neprázdného audio frame (rozpoznávač běží a dostává audio)

8) re-registrace - vyžádaná SIP ústřednou

9) Zavěšení SIP hovoru / uzavření WS

po ukončení session jsou zneplatněny SIP přihlašovací údaje a worker je zrestartován.

SpeechCloud.js

JavaScript knihovna

```
// Sestavení parametrů
var options = {
    uri: SPEECHCLOUD_URI,
    // audio výstup do HTML5 audio elementu
    tts: true || false || 'selector_audio_elementu'
}
// Vytvoření instance SpeechCloud
var speechCloud = new SpeechCloud(options);
// Propojení SIP a WS
speechCloud.init();
```

Použití SpeechCloud.js

```
// Registrace handleru události
speechCloud.on('asr_result', function (event) {
    // handle event - asr result
});
// Volání metody
speechCloud.asr_recognize();
// Volání metody s parametry
speechCloud.tts_synthesize({text: 'text_to_synthesis',
voice: 'voice_name'});
```

Pozn.: Validace JSON Schema probíhá na workeru při přijmutí zprávy (metody) a před odesláním události

SpeechCloud.py

Python knihovna

```
from speechcloud import SpeechCloud
# Vytvoření instance SpeechCloud
speechcloud = SpeechCloud(SPEECHCLOUD_URI)

# Propojení SIP a WS
speechcloud.init()

# Ukončení SIP a WS
speechcloud.terminate()
```

Použití SpeechCloud.py

```
# Registrace handleru události
def result(result, partial_result, **other):
    …

speechcloud.on('asr_result', result);

# Volání metody
speechcloud.asr_recognize();

# Volání metody s parametry
speechcloud.tts_synthesize(text='text_to_synthesis');
```

Použití SpeechCloud.py

```
# Hlavní smyčka

for msg in speechcloud.run_async(timeout=1.):
    if msg is None:
        # vypršel timeout (1 sekunda)
        continue

    # Vyzvednutí typu zprávy
    msg_type = msg.pop('type')

    if msg_type == 'asr_result':
        # Obsluha zprávy
        print msg['result']
```

Klíčové metody a události

Metody

- asr_pauze
- asr_recognize
- asr_process_text
- slu_set_grammars
- tts_synthesize
- tts_stop

Události

- sc_start_session
- asr_ready
- asr_paused
- asr_recognizing
- asr_signal
- asr_result
- slu_entities
- tts_started
- tts_done
- asr_audio_record

Začátek a konec rozpoznávání

`asr_pause()`, `asr_recognize()`
zapínají/vypínají propouštění audia do ASR

pro ASR `asr_pause()` znamená “konec věty”
RTP streamy proudí i při zastaveném ASR

při běžícím ASR informace o energii signálu v událostech `asr_signal`

Zpracování textu

`asr_process_text(text: string/array)`

Nařeže `text` podle bílých znaků (pokud není pole) a zpracuje stejně jako výstup z ASR.

Lze očekávat události `asr_result` a `slu_entities`

Vhodné k testování nebo pro porozumění textu

Registrace gramatik pro SED

`slu_set_grammars(grammars: array)`

Nastaví seznam gramatik, které se budou detekovat v ASR výsledku.

`entity: string`

Jméno sémantické entity (viz `slu_entities`)

`type: enum`

esgf/abnf/xml - odkaz pomocí URI

*-inline - v data přímo obsah gramatiky

`data: URI/data`

odkaz nebo samotná gramatiková data

Syntéza řeči z textu

`tts_synthesize(text: string, voice: string)`

Sesyntetizuje řetězec `text` a přehraje ho do RTP streamu.

Je-li zavoláno vícekrát, pak se promluvy bufferují.

`tts_stop()`

Zastaví přehrávání do RTP streamu.

Pokud je nabufferováno více promluv, zastaví všechny.

`tts_started, tts_done`

Události vygenerované na začátku syntézy, resp. po jejím skončení.

Aktivace session a příchod audio

`sc_start_session`

Speciální událost vznikající po sestavení session

`session_id` - Identifikátor session

`session_uri` - URI, kde je dostupný log ze session

`schema` - JSON schéma pro všechny události a metody

`schema_uri` - URI, kde je JSON schéma komunikace

`asr_ready`

Událost vygenerovaná po sestavení session a příchodu prvního nenulového audio frame.

V tomto stavu je spojen SIP, WS i RTP

Session a schema URI

`format` - lze si říci o různé formáty výstupu

- `format=json` - standardní JSON formát
- `format=yaml` - YAML formát
- `format=json.html` - HTML formátovaný JSON
- `format=yaml.html` - HTML formátovaný YAML
- `format=docson` - Interaktivní dokumentace (jen Schema)

`session_uri` - URI, kde je dostupný log ze session

např.: https://cak.zcu.cz:9443/v1/session/YoZYDrRQLtRYzq2hXC4wj5?format=yaml.html

`schema_uri` - URI, kde je JSON schéma komunikace

např.: https://cak.zcu.cz:9443/v1/schema/wss/numbers/?format=docson

Stav rozpoznávače

`asr_paused`

`asr_recognizing`

Indikují stav pozastaveno/rozpoznáváno

`asr_signal`

Zpětná vazba o energii audio signálu

`speech : bool` - flag je/není řeč

`level : number` - úroveň energie, ideálně z intervalu cca (3.0 - 5.5)

Výsledek rozpoznávání

`asr_result`

ASR vrátilo výsledek rozpoznávání

`result: string` - výsledek rozpoznávání

další atributy závislé na SpeechCloud aplikaci

mohou následovat další události z dalších nástrojů

např. `slu_entities`

Výsledek porozumění

`slu_entities`

Výsledek detekce sémantických entit

`entities` - pole objektů

`prob` - pravděpodobnost

`values` - pole sémantických entit (string)

`classes` - sémantické entity agregované podle jména

`segments` - časově uspořádané sémantické entity

slu_entities.entities


```
"entities": [ {
  "values": ["CS:CSA:0:2:4:CSA024", "CMD:climb", "FL:1:8:0"],
  "prob": 0.94863188564768
}, {
  "values": ["CS:CSA:0:2:4:CSA024", "CMD:climb", "FL:1:6:0"],
  "prob": 0.051167212766021
}, {
  "values": ["CS:CSA:0:2:4:CSA024", "CMD:climb"],
  "prob": 0.00020090158629558
} ]
```


`CS`, `CMD` a `FL` jsou jména sémantických entit

slu_entities.classes


```
"classes": {
    "CS": [
      [ "CSA:0:2:4:CSA024", 1 ]
    ],
    "CMD": [
      [ "climb", 1 ]
    ],
    "FL": [
      [ "1:8:0", 0.948631885647680 ],
      [ "1:6:0", 0.051167212766021 ]
    ]
}
```

slu_entities.segments


```
"segments": [
[
   [ 0, 1 ], { "CS:CSA:0:2:4:CSA024": 1 }
],
[
   [ 1, 2 ], { "CMD:climb": 1 }
],
[
   [ 2, 3 ], { "FL:1:8:0": 0.948631885647 68,
               "FL:1:6:0": 0.051167212766021 }
]
]
```

SLU gramatiky

Vychází ze Speech Recognition Grammar Specification Version 1.0 (SRGS)

http://www.w3.org/TR/speech-grammar/

- ABNF zápis
  - textový, lze převést na XML
- XML zápis

Popisují pouze sémantické entity, nikoli výplňová slova

Příklady SLU gramatik

Z projektu IT-BLP (řízení letového provozu)

http://itblp.zcu.cz/app-demo/atg1/static/grms/

ALT.abnf	CMD.abnf	CS.abnf
FL.abnf	FR.abnf	HE.abnf
PO.abnf	QNF.abnf	RA.abnf
SP.abnf	SQ.abnf	TU.abnf
TWR.abnf		

FL.abnf

```
#ABNF 1.0 UTF-8 cs;
grammar FL;
root $FL;

public $FL = (
   ( (FL|level) [is]
     ([$digit09] $digit09 $digit09 | $digit09 $hundred) ) |
   ( ([$digit09] $digit09 $digit09 | $digit09 $hundred) )
) [(or below {below} | or above {above}] ;

$digit09 = ( &0 {0} | &1 {1} | &2 {2} | &3 {3} | &4 {4}
| &5 {5} | &6 {6} | &7 {7} | &8 {8} | &9 {9} );

$hundred = (hundred {0} {0});
```

level is &2 hundred or below ⇒ {2} {0} {0} {below} ⇒ **FL:2:0:0:below**

ABNF gramatiky

definice pravidla	`$FL = … ;`
terminál	`hundred`
řetězení terminálů	`or below`
odkaz na pravidlo	`$digit09`
alternativy	`FL | level`
opakování	`jedno a více+` `nula a více*`
volitelná část	`[is]`
tag	`{1}`

Nastavení SLU gramatik

Možno použít více entit, každá dána gramatikou.

Gramatiky typu ESGF, **ABNF**, XML

Stahují se z **URI** nebo jsou **inline**

Metoda `slu_set_grammars`

Potvrzení událostí `slu_set_grammars_done`

Při chybě událost `sc_error`

Nastavení SLU gramatik

```
# Definice inline gramatiky (Python)
grammar = u'''#ABNF 1.0 UTF-8 cs;
grammar numbers;

root $NUM;
public $NUM  = (
    jedna {1}  |
    (dva | dvě) {2} |
    tři {3} |
    čtyři {4} |
    pět {5} |
    šest {6} |
    (sedm | sedum) {7} |
    (osm | osum) {8} |
    devět {9}
);
'''
```

Nastavení SLU gramatik

```
# Registrace gramatiky
speechcloud.slu_set_grammars(grammars=[{'data': grammar,
                                        'entity': 'NUM',
                                        'type': 'abnf-inline'} ])

# Definice handleru
def handle_entities(**msg):
    entities = msg['entities']
    ...

# Registrace handleru
speechcloud.on('slu_entities', handle_entities)
```