

УТВЕРЖДЕН

ДССЛ.00103-01 31 01 - ЛУ

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

«3i Speech Detector SDK»

Описание применения

ДССЛ.00103-01 31 01

Листов 22

Литера О₁

2016

АННОТАЦИЯ

Настоящий документ предназначен для ознакомления с программным обеспечением «3i Speech Detector SDK» и содержит описание интерфейса программирования (API) для программистов, обеспечивающих использование 3i Speech Detector SDK в качестве модуля, встраиваемого в другое программное обеспечение.

В разделе 1 приводятся сведения о назначении и составе 3i Speech Detector SDK.

В разделе 2 указаны требования к программно-техническим средствам, необходимым для работы 3i Speech Detector SDK.

В разделе 3 указывается описание задач, решаемых 3i Speech Detector SDK, даются сведения об используемых технологиях.

В разделе 4 даются сведения об установке 3i Speech Detector SDK.

В разделе 5 приводится описание интерфейса программирования 3i Speech Detector SDK.

В разделе 6 даются сведения о входных и выходных данных 3i Speech Detector SDK.

В разделе 7 приводятся основные сообщения оператору при работе с 3i Speech Detector SDK.

СОДЕРЖАНИЕ

АННОТАЦИЯ	3
1. Назначение программы	5
2. Условия применения	6
3. Описание задачи	7
3.1. Технологии работы с речевыми данными	7
3.2. Состав 3i Speech Detector SDK	8
4. Вызов и загрузка программы	9
4.1. Установка 3i Speech Detector SDK	9
5. Выполнение программы	10
5.1. Комплект 3i Speech Detector SDK	10
5.2. Описание функционала API	10
6. Входные и выходные данные	17
7. Сообщения оператору	18
Перечень сокращений	20

1. НАЗНАЧЕНИЕ ПРОГРАММЫ

Программное обеспечение «3i Speech Detector SDK» предназначено для решения определения в потоке аудиосигнала речевых участков, содержащих функциональные возможности преобразования речь-текст (для различных каналов и источников речи и языков). 3i Speech Detector SDK используется в качестве библиотеки модулей, встраиваемых в другое программное обеспечение, предоставляя разработчику соответствующий функционал API.

API (сокр. англ. Application Programming Interface, интерфейс программирования приложений, интерфейс прикладного программирования) – набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением для использования во внешних программных продуктах.

К основным преимуществам приложения, предоставляющего shared object API в качестве инструмента доступа к функциональным возможностям, можно отнести:

- надёжность (за счет отсутствия необходимости взаимодействия с удаленным модулем);
- производительность (за счет использования кэша и подгрузки кода и данных в пространство адресов приложения);
- прозрачность системы взаимодействия;
- легкость внесения изменений;
- масштабируемость.

Продукт реализован в виде динамической библиотеки (so), написанной на языке C++.

2. УСЛОВИЯ ПРИМЕНЕНИЯ

2.1. Для функционирования 3i Speech Detector SDK необходима вычислительная система с параметрами не хуже:

- CPU Intel Core i7 – 5820K 3.3 ГГц (6 физических вычислительных ядер);
- ОЗУ 16 ГБ;
- 100 Гб свободного места на жёстком диске.

2.2. Для функционирования 3i Speech Detector SDK на вычислительной системе должно быть установлено следующее общее программное обеспечение:

для семейства операционных систем Microsoft Windows:

- операционная система – Microsoft Windows 7 (x64) SP1 или выше;
- распространяемый пакет Microsoft VC 2013 Redist (x64).

для семейства 64-разрядных операционных систем на базе ядра Linux:

- операционная система – представитель семейства 64-разрядных операционных систем на базе ядра Linux (CentOS, Ubuntu, Astra Linux и пр.);
- распространяемый пакет libtbb2 для семейства операционных систем на базе ядра Linux, обеспечивающий управление нагрузкой многоядерных процессоров при исполнении модулей C++ (параллельная обработка данных).

3. ОПИСАНИЕ ЗАДАЧИ

3.1. Технологии работы с речевыми данными

Возможности модуля 3i Speech Detector SDK определения (детектирования) речи в аудиосигнале базируются на технологиях разметки аудиосигнала на речь, тоны и прочие звуковые сегменты.

Алгоритм обнаружения сигналов гудков и тонального набора анализирует информацию о стационарности частот в спектре сигнала. Однотоновые гудки обнаруживаются при стационарности частотного максимума на протяжении не менее 120мс в полосе выше 300Гц, двухтоновые – не менее 80мс. Если двухтоновый гудок является DTMF-сигналом, то производится определение нажатой клавиши или типа сигнала ответа АТС.

Обнаружение участков тишины, фонового шума и речевой активности выполняется с использованием смесей гауссовых распределений (VAD). Технология основана на анализе информации об энергии во входном сигнале. На каждом шаге вычисляется энергия сигнала в частотной полосе 100-3600Гц. Участки, энергия которых меньше порогового значения 0.04 (значение определено эмпирически) отмечаются, как участки с тишиной. С использованием одномерных смесей гауссовых распределений строится классификатор «фон»/«речевая активность», с помощью которого размечаются сегменты входного звукового потока, не относящиеся к тишине, гудкам и DTMF.

Обнаружение участков речевой активности выполняется с помощью нейронной сети. Нейронные сети – мощный инструмент для моделирования и последующего распознавания многопараметрических объектов. Применимо к задаче обнаружения участков речевой активности нейронная сеть используется для определения вероятности того, что в анализируемом фрагменте аудио записи присутствует речь. Используемая нейронная сеть принимает на вход 11 векторов признаков, имеет 3 скрытых слоя и выходной softmax-слой на 2 нейрона («речь» и «не_речь»). В текущей версии реализована классификация трёх типов: «речь»,

«музыка» и «фон/шум». Работа сети оптимизирована с использованием SSE-инструкций и может быть распределена по ядрам процессора, что позволяет достичь максимальной производительности программного модуля.

3.2. Состав 3i Speech Detector SDK

3i Speech Detector SDK включает следующие модули (библиотеки):

- модель детектирования речи в аудиосигнале.

4. ВЫЗОВ И ЗАГРУЗКА ПРОГРАММЫ

4.1. Установка 3i Speech Detector SDK

Установка модулей (библиотек) 3i Speech Detector SDK осуществляется переносом (копированием) архива с модулями с загрузочного диска в требуемую директорию на жестком диске вычислительной системы пользователя. После переноса (копирования) архива модулей, динамические библиотеки (so) модулей разархивируются из него в требуемую директорию на жестком диске вычислительной системы пользователя. Модули (библиотеки) 3i Speech Detector SDK готовы к эксплуатации.

5. ВЫПОЛНЕНИЕ ПРОГРАММЫ

5.1. Комплект 3i Speech Detector SDK

Модули 3i Speech Detector SDK имеют классический интерфейс, характерный для динамически подгружаемых библиотек (so). В комплекте с модулем поставляется:

- бинарный файл динамически связываемой библиотеки (расширение so);
- бинарный файл прокси-библиотеки для статического связывания приложения пользователя с динамически связываемой библиотекой (расширение a);
- файл-хидер, содержащий исходный код интерфейса модуля на C++ для сборки связывания динамической библиотеки модуля и приложения пользователя (расширение h);
- модели обработки данных.

Описание функционала API 3i Speech Detector SDK представлено ниже.

5.2. Описание функционала API

Список функций API 3i Speech Detector SDK представлен в таблице 5.2.1

Таблица 5.2.1 Список функций API 3i Speech Detector SDK

Наименование функции	Описание функции
<i>sd_version</i>	Информация о версии библиотеки
<i>sd_init</i> <i>sd_deinit</i>	Начало/окончание работы с библиотекой
<i>sd_is_init</i>	Проверка статуса инициализации библиотеки
<i>sd_get_err_msg</i>	Получение информации об ошибках

Наименование функции	Описание функции
<i>sd_process_buf</i>	Получение «сырой» разметки
<i>sd_get_segm_vad</i> <i>sd_get_segm_dnn</i>	Получение «итоговой» разметки
<i>sd_get_segm_smooth</i>	Сглаживание разметки
<i>sd_export_segm_to_bin</i> <i>sd_export_segm_to_bin_simple</i>	Экспорт разметки в бинарный файл
<i>sd_destroy_raw_data</i>	Удаление разметки (очистка памяти)

Описание функций API 3i Speech Detector SDK представлено ниже.

1. Информация о версии библиотеки

API функции:

```
char const* sd_version();
```

Функция возвращает строку вида “<MAJOR>.<MINOR>.<PATCH>”

2. Начало/окончание работы с библиотекой

Перед началом работы с другими функциями библиотеки, необходимо инициализировать её, задав параметры работы детекторов.

Функция инициализации библиотеки имеет вид:

```
int sd_init(char const* model_dir_path, bool dnn_mt_mode);
```

Параметры:

model_dir_path – путь к каталогу с конфигурационными файлами и моделями;

dnn_mt_mode – флаг режима работы нейросетевого детектора речи (TRUE – многопоточный режим; FALSE – однопоточный режим).

В случае успешной загрузки параметров предобработки сигнала и нейросетевой модели будет возвращен «0», иначе – код ошибки.

По окончании работы с библиотекой следует вызвать функцию её деинициализации:

```
void sd_deinit();
```

Функция `sd_deinit()` будет «ожидать» окончания работы ранее вызванных функций библиотеки и не завершивших свою работу на момент запуска этой функции. В процессе деинициализации функции библиотеки будут «заблокированы». Обе функции выполняются в однопоточном режиме.

3. Проверка статуса инициализации библиотеки

API функции:

```
bool sd_is_init();
```

Функция возвращает TRUE в случае, когда библиотека готова к работе и не находится в состоянии деинициализации.

4. Получение информации об ошибках

Большинство функций библиотеки возвращают численное значение кода завершения операции. Текстовую (более информативную) интерпретацию кода можно получить с помощью функции:

```
const char* sd_get_err_msg(int code);
```

Параметры:

code – код завершения операции.

5. Детектирование речи

В процессе обработки аудио данных можно выделить несколько этапов.

1) Получение «сырой» разметки.

Источником данных может служить WAV-файл или буфер памяти, соответствующие требованиям к входным данным. На выходе пользователь получает void-указатель на «сырую» разметку (в сущности, под этим указателем «скрывается» матрица результатов работы каждого детектора в отдельности, а также дополнительная информация о параметрах их работы).

2) Получение «итоговой» разметки «речь/не_речь».

«Сырая» разметка может быть скомпонована в «итоговую» несколькими способами: с определением речевых сегментов по результатам VAD или по результатам DNN-детектора.

3) Сглаживание «итоговой» разметки.

4) Экспорт разметки в файл (при необходимости).

Возможен экспорт разметки в полном и в упрощённом варианте.

«Полная» разметка – в выходной файл записываются коды сегментов всех обнаруженных классов сигналов.

«Упрощённая» разметка – в выходной файл записываются коды только «речевых» сегментов (код «128»), остальные помечаются кодом «0».

5) Удаление данных, полученных на этапах 1 и 2.

В дальнейшем при описании API функций будут использоваться следующие определения:

```
typedef void* raw_data_ptr;           // указатель на «сырую» разметку
typedef unsigned char segm_data_t;   // базовый тип для хранения метки
                                     // класса
typedef segm_data_t* segm_data_ptr;  // указатель на массив сегментов
```

5.1 Получение «сырой» разметки

По данным их буфера памяти.

API функции:

```
raw_data_ptr sd_process_buf(int & res_code, short const*
audio_data_ptr, int audio_data_size);
```

Параметры:

res_code – адрес переменной, куда будет записан код завершения операции;

audio_data_ptr – указатель на массив с отсчетами звукового сигнала;

audio_data_size – размер звукового буфера (количество отсчётов).

По данным из WAV-файла.

API функции:

```
raw_data_ptr sd_process_buf(int & res_code, char const* wav_fpath);
```

Параметры:

res_code – адрес переменной, куда будет записан код завершения операции;

wav_fpath – путь к WAV-файлу с аудио данными.

В случае успеха функции записывают указатель на массив кодов сегментов, а в *res_code* записывают код «0», иначе – в *res_code* будет записан код ошибки.

Иногда возникают ситуации, когда на вход функций поступает звуковой буфер, в котором содержится достаточно малое количество речи и фонового шума. В таких случаях VAD'ом может быть возвращен код ошибки CLUSTERING_ERROR. Это означает, что детектор не смог построить классификатор «фон-речь» на данном звуковом буфере.

5.2 Получение «итоговой» разметки

API функций:

```
segm_data_ptr sd_get_segm_vad( int & res_code, int & out_segm_size,  
const raw_data_ptr raw_data);
```

```
segm_data_ptr sd_get_segm_dnn( int & res_code, int & out_segm_size,  
const raw_data_ptr raw_data);
```

Параметры:

res_code – адрес переменной, куда будет записан код завершения операции;

out_segm_size – адрес переменной для записи размера выходного массива кодов сегментов;

raw_data – указатель на «сырую» разметку.

5.3 Сглаживание разметки

API функции:

```
segm_data_ptr sd_get_segm_smooth(int & res_code, const segm_data_ptr  
segm_data_ptr, int segm_data_size);
```

Параметры:

res_code – адрес переменной, куда будет записан код завершения операции;

segm_data_ptr – указатель на «итоговую» разметку, которую необходимо «сгладить»;

segm_data_size – размер входной разметки.

5.4 Экспорт разметки в бинарный файл

Разметка, возвращаемая функциями *sd_get_segm_vad*, *sd_get_segm_dnn*, *sd_get_segm_smooth* может быть сохранена в бинарный файл с помощью отдельной функции.

API функций:

```
int sd_export_segm_to_bin(const segm_data_ptr segm_data_ptr,  
int segm_data_size, const char* dest_fpath);  
int sd_export_segm_to_bin_simple(const segm_data_ptr segm_data_ptr,  
int segm_data_size, const char* dest_fpath);
```

Параметры:

segm_data_ptr – указатель на сохраняемую разметку;

segm_data_size – количество сегментов;

dest_fpath – путь к выходному файлу.

Формат выходного файла:

[int32] [byte][byte] ... [byte], (a) – количество кодов, (b) – последовательность кодов.
(a) (b)

В случае успеха функции возвращают код «0», иначе — код ошибки.

5.5 Удаление разметки (очистка памяти)

Очистка и высвобождение памяти, выделенной для хранения «сырой» разметки.

API функции:

```
void sd_destroy_raw_data(raw_data_ptr & data_ptr);
```

Очистка и высвобождение памяти, выделенной для хранения «итоговой» разметки.

API функции:

```
void sd_destroy_segm_data(segm_data_ptr & data_ptr);
```

Параметры:

data_ptr – ссылка на указатель на разметку, которая подлежит удалению.

6. ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

Входные данные:

Звуковые данные, поступающие на вход функциям библиотеки, могут храниться в буфере памяти или в бинарном файле.

Требования к буферу памяти, в котором хранятся аудио данные:

- кодирование: несжатый WAV-PCM;
- квантование отсчёта: 16 бит;
- частота дискретизации: 8000 Гц.

Требование к аудио файлам:

- кодирование: несжатый WAV-PCM, A-Law или Mu-Law;
- квантование отсчёта: 16 бит (при кодировании «несжатый WAV-PCM»);
- частота дискретизации: 8 кГц;
- тип файла: *.wav.

Выходные данные: указатель на массив кодов сегментов.

7. СООБЩЕНИЯ ОПЕРАТОРУ

Все сообщения оператору реализуются через коды возвращаемых значений функциями модуля, далее дается расшифровка этих значений.

Таблица 7.1 Список кодов сегментов

Код	Значение
0	Тип сегмента не задан или не определен
1	Тишина
2	Фоновый шум
4	Однотоновый гудок
5	Двухтоновый гудок
100	DTMF: клавиша «0»
101	DTMF: клавиша «1»
102	DTMF: клавиша «2»
103	DTMF: клавиша «3»
104	DTMF: клавиша «4»
105	DTMF: клавиша «5»
106	DTMF: клавиша «6»
107	DTMF: клавиша «7»
108	DTMF: клавиша «8»
109	DTMF: клавиша «9»
110	DTMF: клавиша «А»
111	DTMF: клавиша «В»
112	DTMF: клавиша «С»
113	DTMF: клавиша «D»
114	DTMF: клавиша «*» (asterisk)
115	DTMF: клавиша «#» (sharp)
116	DTMF: «готовность» (readiness)

Код	Значение
117	DTMF: «занято» (busy)
118	DTMF: «ringback» (US)
128	Речь

Таблица 7.2.2 Список возвращаемых кодов

Код	Значение
0	NO_ERR
1	UNKNOWN_ERROR
3	MEMORY_ALLOCATION_ERROR
5	INCORRECT_INPUT_PARAM
7	INCORRECT_INPUT_CONTAINER
8	INCORRECT_INIT_PARAMS
16	IS_NOT_INIT
17	ALREADY_INIT
36	UNABLE_TO_SAVE_FILE
37	UNABLE_TO_OPEN_FILE
38	UNABLE_TO_LOAD_FILE
39	INCORRECT_FILE_FORMAT
42	UNABLE_TO_CALC_FEATURES
43	UNABLE_TO_CALC_SPECTRAL_COEFS
44	UNABLE_TO_CALC_SEGMENTATION
54	UNABLE_TO_SAVE_MODEL
55	UNABLE_TO_LOAD_MODEL
64	CLUSTERING_ERROR
46	UNABLE_TO_CALC_ENERGY
88	ANN_UNABLE_TO_LOAD

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

В настоящем документе приняты следующие условные обозначения:

ПО	Программное обеспечение
3i 3i Speech Detector SDK	ПО «3i Speech Detector SDK»
API	сокр. англ. Application Programming Interface, интерфейс программирования приложений – набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением для использования во внешних программных продуктах
CPU	сокр. англ. Central Processing Unit – электронный блок либо интегральная схема (микропроцессор), исполняющая машинные инструкции (код программ).
DNN	сокр. англ. Deep Neural Network, искусственная нейронная сеть с несколькими скрытыми слоями.
DTMF	сокр. англ. Dual-Tone Multi-Frequency, двухтональный многочастотный аналоговый сигнал, используемый для набора телефонного номера.
GMM	сокр. англ. Gaussian Mixture Model – статистическая модель плотности вероятности, выраженная суммой нормальных многомерных распределений.
MFCC	сокр. англ. Mel-Frequency Cepstrum Coefficients – представление кратковременного спектра, основанное на линейном косинусном преобразовании логарифмированного амплитудного спектра по мел-частотной шкале.
PCA	сокр. англ. Principal Component Analysis – метод

	сокращения размерности статистических данных.
PCM	сокр. англ. Pulse Code Modulation, импульсно-кодовая модуляция, термин применяется в смысле типа кодирования аудио-сигнала.
SDK	сокр. англ. Source Development Kit – комплект средств разработки, который позволяет специалистам по программному обеспечению создавать приложения.
SO	сокр. англ. Shared object – динамическая библиотека в ОС на базе ядра Linux, позволяющая многократное использование различными программными приложениями.
UBM	сокр. англ. Universal Background Model – статистическая модель пространства признаков голоса.
WFST	сокр. англ. Weighted Finite State Transducer, взвешенный конечно-автоматный преобразователь, автомат Мили со взвешенными дугами.

