УТВЕРЖДЕН

ДССЛ.00101-01 31 01 - ЛУ

СПЕЦИАЛЬНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ
«3i Speaker ID SDK»

Описание применения

ДССЛ.00101-01 31 01

Листов 28

Литера $О_1$

2015

АННОТАЦИЯ

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

В разделе 1 приводятся сведения о назначении и составе 3i Speaker ID SDK.

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

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

В разделе 4 даются сведения об установке 3i Speaker ID SDK.

В разделе 5 приводится описание интерфейса программирования 3i Speaker ID SDK.

В разделе 6 даются сведения о входных и выходных данных 3i Speaker ID SDK.

В разделе 7 приводятся основные сообщения оператору при работе с 3i Speaker ID SDK.

По всем вопросам, связанным с использованием СПО «3i Speaker ID SDK» можно обращаться по электронной почте support@dss-lab.ru или по телефону +7 (495) 645-44-70 по будним дням с 10 до 18 часов, время московское.

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 Speaker ID SDK 9
4. Вызов и загрузка программы 10
4.1 Установка 3i Speaker ID SDK 10
4.2 Получение лицензионного файла-ключа 10
5. Выполнение программы 12
5.1 Комплект модуля 3i Speaker ID SDK 12
5.2 Описание функционала API 12
6. Входные и выходные данные 25
7. Сообщения оператору 26
Перечень сокращений 27

1. НАЗНАЧЕНИЕ ПРОГРАММЫ

Специальное программное обеспечение «3i Speaker ID SDK» предназначено для решения задач распознавания речи, текстонезависимой идентификации личности диктора по голосу. 3i Speaker ID SDK используется в качестве программного обеспечения, 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 DLL 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 (DLL), написанной на языке C++. Допускается использование в нескольких параллельных потоках.

2. УСЛОВИЯ ПРИМЕНЕНИЯ

2.1. Для функционирования 3i Speaker ID 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 Speaker ID SDK на вычислительной системе должно быть установлено следующее общее программное обеспечение:

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

3. ОПИСАНИЕ ЗАДАЧИ

3.1 Технологии работы с речевыми данными

Технология идентификации дикторов по голосу основана на последних достижениях в области анализа речи и принятия решений. В предлагаемом продукте реализован целый ряд самостоятельных методов голосовой идентификации. Среди них известные и уже ставшие традиционными подходы, основанные на Гауссовых смесях (GMM – Gaussian Mixture Model), собственных векторах (i-vectors) и супервекторах. Кроме традиционных методов, в библиотеке применены оригинальные решения как в области первичной обработки речевых сигналов и кодирования акустических признаков речи, так и в области построения голосовых моделей и методов принятия решений. Благодаря этому достигнута высокая устойчивость правильной идентификации в различных каналах связи.

Встроенный специальный модуль, выполняющий очистку входного звукового потока от посторонних – неречевых – вставок и выделяющий участки с речевой активностью на основе анализа энергии сигнала, способствует увеличению качества идентификации.

Качество идентификации

Тестирование технологии проводилось и использованием мультиязычного речевого корпуса, из которого было отобрано 50 «целевых» («своих») дикторов женского и мужского пола в равных долях и 2982 «не целевых» («чужих») дикторов.

На рисунках 2.1 - 2.3 представлены графики значений ошибок первого и второго рода (FRR и FAR соответственно) в условиях работы в телефонном (phone) и телерадиовещательном каналах связи (broadcast).

FAR (False Alarm Ratio) – ошибка ложного срабатывания («чужой» диктор принят за «своего», либо диктор из числа «своих» был перепутан с иным «целевым» диктором).

FRR (False Rejection Ratio) – ошибка ложного пропуска «своего» диктора.

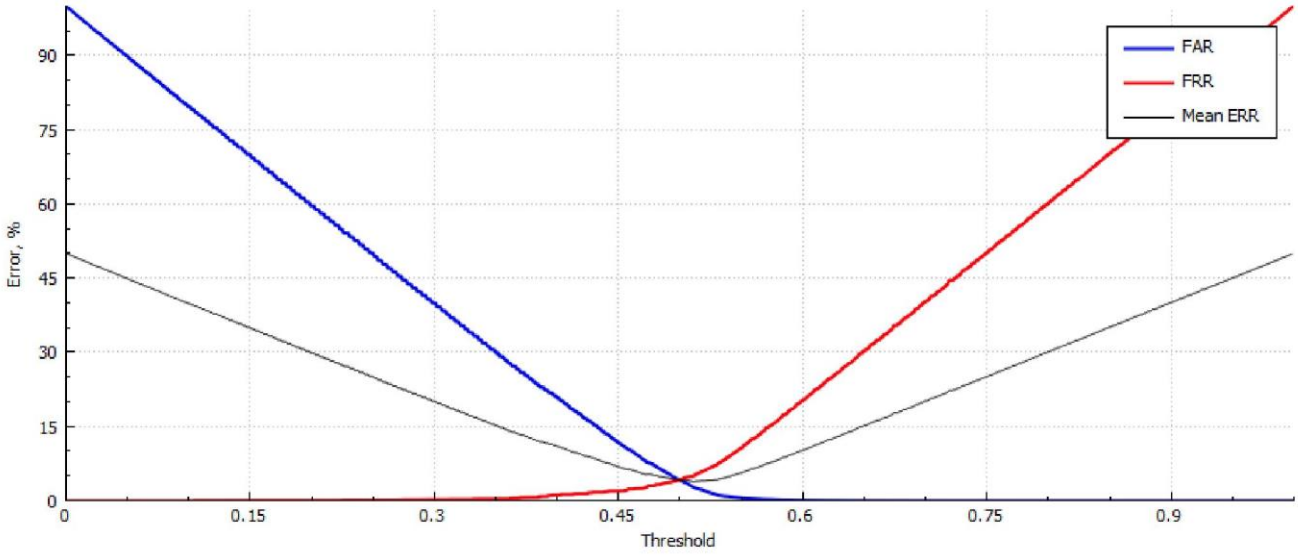

Рисунок 2.1 Графики ошибок первого и второго рода (phone)

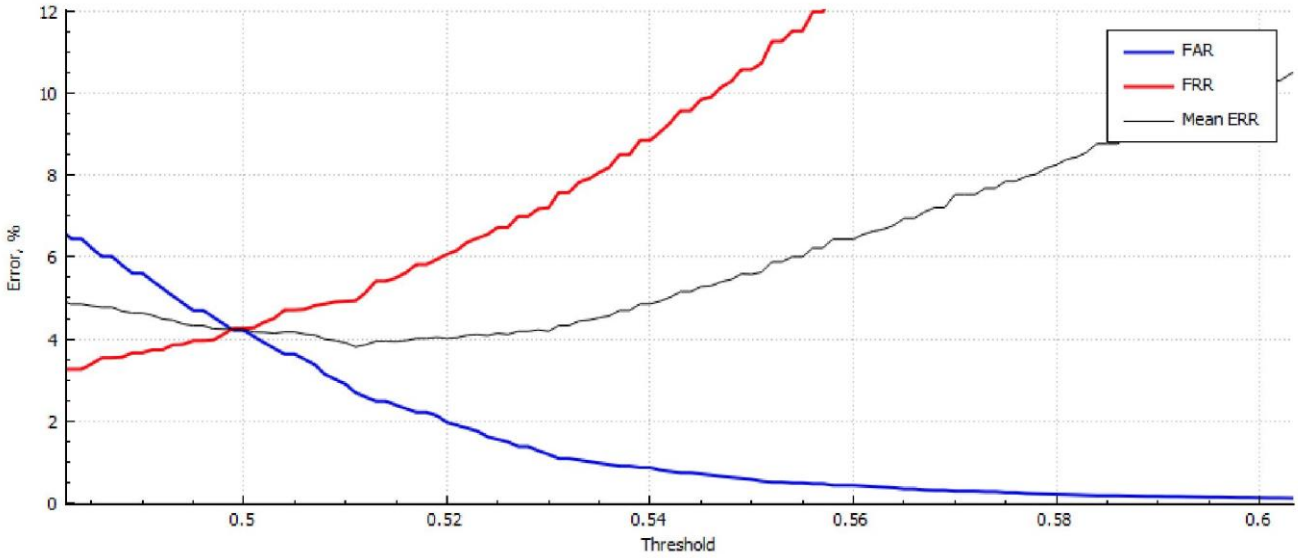

Рисунок 2.2 Графики ошибок первого и второго рода (phone)

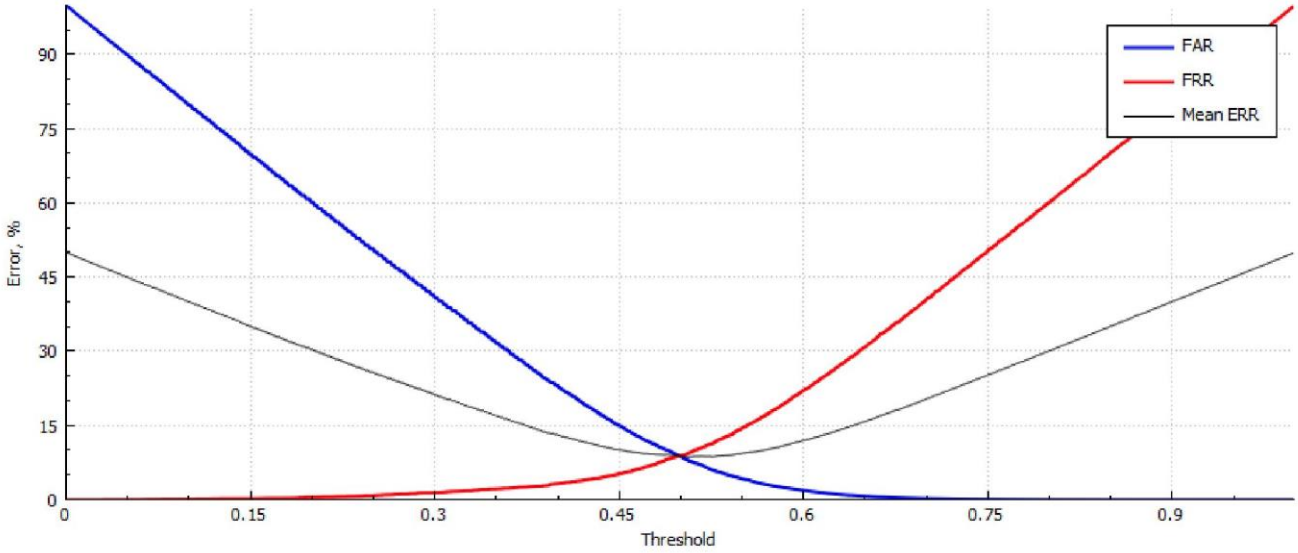

Рисунок 2.3 Графики ошибок первого и второго рода (broadcast)

3.2 Состав 3i Speaker ID SDK

3i Speaker ID SDK – модуль проводит определение диктора (идентификация диктора).

4. ВЫЗОВ И ЗАГРУЗКА ПРОГРАММЫ

4.1 Установка 3i Speaker ID SDK

Установка модуля 3i Speaker ID SDK осуществляется переносом (копированием) архивов с модулями с загрузочного диска в требуемую директорию на жестком диске сервера. После переноса (копирования) архива модуля динамические библиотеки (DLL) модуля 3i Speaker ID SDK разархивируются из него в требуемую директорию на жестком диске сервера. Модуль готов к эксплуатации.

4.2 Получение лицензионного файла-ключа

Бинарные файлы модулей 3i Speaker ID SDK защищены от нелицензионного использования и копирования, лицензионное использование предполагает получение файла-ключа, который специфичен для системы пользователя и бинарного файла модуля. Таким образом, полученный файл-ключ нельзя использовать на другой машине или для другого модуля, что исключает нелицензионное копирование и использование модулей. Файл-ключ может быть получен у технической поддержки разработчика в случае наличия у пользователя лицензии. Процедура получения файла-ключа такова:

1) В папке каждого модуля находится директория GenHardID, содержащая исполняемый файл GenHardID_console.exe, генерирующая уникальный идентификационный код системы пользователя. После распаковки архива модуля следует запустить данный файл.
2) Результатом работы GenHardID_console.exe является файл hardware_id.hid, содержащий идентификационный код системы пользователя. Это код одинаков для любого модуля и характеризует систему (машину) пользователя, таким образом, допустимо получить этот файл один раз. Файл hardware_id.hid появляется в той же папке, в которой находится исполняемый файл GenHardID_console.exe.

3) Файл идентификационного кода системы hardware_id.hid следует передать в техническую поддержку 3i, сопровождая информацией о лицензии.
4) При подтверждении пользовательской лицензии, техническая поддержка высылает файл-ключ пользователю. Файл-ключ уникален для каждого исполняемого файла (библиотеки, DLL). Его следует положить рядом с тем бинарным файлом (исполняемым модулем, DLL), для которого файл-ключ был сгенерирован. После этого, модуль будет способен запуститься и работать.

5. ВЫПОЛНЕНИЕ ПРОГРАММЫ

5.1 Комплект модуля 3i Speaker ID SDK

Модуль 3i Speaker ID SDK имеет классический интерфейс, характерный для динамически подгружаемых библиотек (dll). В комплекте с модулем поставляется:

- бинарный файл динамически связываемой библиотеки (расширение dll);
- бинарный файл прокси-библиотеки для статического связывания приложения пользователя с динамически связываемой библиотекой (расширение lib);
- файл-хидер, содержащий исходный код интерфейса модуля на C++ для сборки связывания динамической библиотеки модуля и приложения пользователя (расширение h);
- модель – бинарный файл с универсальной фоновой моделью (расширение gmm или dgmm), бинарный файл с данными для получения идентификационных векторов (расширение egn), бинарный файл с данными для вычисления метрики MERR (сокр. англ. Mean Error, средняя ошибка), расширение bin;
- конфигурационный файл с параметрами вычисления акустических признаков (расширение conf);
- конфигурационный файл с параметрами построения модели голоса говорящего и указанием метрики определения схожести моделей (расширение conf).

5.2 Описание функционала API

Список функций API 3i Speaker ID SDK представлен в таблице 5.1.

Таблица 5.1 Список функций API 3i Speaker ID SDK

Наименование функции	**Описание функции**

Наименование функции	Описание функции
init	Инициализация библиотеки
initWithVst	Инициализация библиотеки с включением модуля сегментации диалогов по голосам
delnit	Деинициализация библиотеки
getModel FromBuf *getModel FromFile*	Построения векторной модели голоса диктора («базовые» функции). ***Источник:*** буфер памяти или WAV-файл
getModel FromBufExt *getModel FromFileExt*	Построение векторной модели голоса диктора с возвратом дополнительной информации об обработанном звуковом сигнале: длительность всего сигнала; длительность речевых; сегментов, на которых вычислена модель. ***Источник:*** буфер памяти или WAV-файл
getModelFromDialog FromBufExt *getModelFromDialog FromFileExt*	Построение векторной модели голоса диктора с предварительным разделением диалога по голосам и возвратом дополнительной информации об обработанном сигнале. ***Источник:*** буфер памяти или WAV-файл
saveModel	Сохранение векторной модели голоса диктора в бинарный файл
saveModelExt	Сохранение векторной модели голоса диктора в бинарный файл с указанием дополнительной информации об обработанном сигнале
loadModel	Загрузка векторной модели голоса диктора из бинарного файла

Наименование функции	Описание функции
loadModelExt	Загрузка векторной модели голоса диктора из бинарного файла с получением дополнительной информации об обработанном сигнале
compareModels	Вычисление меры близости между двумя моделями голосов (непосредственно идентификация)
compareModelsExt	Вычисление меры близости между двумя моделями голосов с получением значения «чистого» косинусного расстояния
deleteData	Очистить и удалить контейнер с моделью
deleteModels	Очистка и удаление указателя на список дикторских моделей

Описание функций API 3i Speaker ID SDK представлено ниже.

Дополнительные типы данных

```
// базовый тип для i-вектора
#define spk_model_base_t double
// указатель на i-вектор
#define spk_model_base_ptr spk_model_base_t*
namespace
sid_sdk {
  typedef struct SID_API_CLASS
  spk_model_t {
    spk_model_t(int modelSize = 0, spk_model_base_ptr modelData =
nullptr, int sigDur = 0, int speechDur = 0);
    ~spk_model_t();
    int SpeechDuration; // общая длительность речевых сегментов, на которых
    построена модель (ms)
    int SignalDuration; // длительность аудио сигнала (ms)
    int ModelSize; // размерность i-вектора
```

```
  spk_model_base_ptr Model; // i-вектор }
 spk_model_t; typedef spk_model_t* spk_model_ptr; //
 указатель на модель голоса диктора
 typedef struct SID_API_CLASS
 spk_models_t {
  virtual ~spk_models_t();
  int Count; // количество дикторских моделей
  spk_model_ptr Models = nullptr; // указатель на массив
 дикторских моделей } spk_models_t;
 typedef spk_models_t* spk_models_ptr; // указатель на список моделей
голосов дикторов (для VST) }
```

Инициализация библиотеки

Прежде, чем приступить к работе с библиотекой, необходимо ее инициализировать. В процессе инициализации производится загрузка универсальной фоновой модели (UBM – Universal Background Model) и данных для проекции супер-векторов на главные компоненты (PCA – Principal Component Analysis), а также параметров вычисления акустических признаков и определение метрики.

API функции:

```
int init(char const* pathToFvConfig, char const* pathToTsmConfig, bool
useInternalSD);
```

PathToFvConfig - путь к конфигурационному файлу для вычисления векторов акустических признаков;

PathToTsmConfig - путь к конфигурационному файлу для построения (обучения) дикторских моделей;

UseInternalSD - значение: TRUE – использовать встроенный детектор речи, FALSE – не использовать.

В ситуациях, когда в одной сессии присутствует голос двух дикторов (ведётся диалог), можно проводить построение дикторских моделей, предварительно инициализировав библиотеку на работу с диалогами.

API функции:

```
int initWithVst(char const* pathToFvConfig, char const* pathToTsmConfig, bool useInternalSD, char const* pathToVstDll, char const* pathToVstConfig);
```

PathToFvConfig - путь к конфигурационному файлу для вычисления векторов акустических признаков;

PathToTsmConfig - путь к конфигурационному файлу для построения (обучения) дикторских моделей;

UseInternalSD – значение: TRUE – использовать встроенный детектор речи, FALSE – не использовать;

PathToVstDll - путь к библиотеке 3iVST;

PathToVstConfig - путь к конфигурационному файлу для модуля сегментации диалогов по голосам.

Установка флага *UseInternalSD=FALSE* подразумевает, что входной аудио поток предварительно очищен от неречевых сигналов, таких как тишина, фоновый шум, гудки и DTMF-сигналы. Процесс вычисления модели голоса диктора может ускориться до 5% от времени, затрачиваемого при *UseInternalSD=TRUE*.

Функции возвращают 0 в случае успеха или код ошибки.

Деинициализация библиотеки

По окончании работы с библиотекой необходимо выгрузить UBM и данные для PCA, загруженные при инициализации библиотеки.

API функции:

```
void deInit();
```

После вызова данной функции будет заблокирована возможность работы с функциями построения дикторских моделей: *getModel_FromFile(),getModel_FromFileExt(), getModelFromDialog_FromFileExt(), getModel_FromBuf(), getModel_FromBufExt(), getModelFromDialog_FromBufExt()*.

Функция *deInit()* будет «ждать» окончания работы всех запущенных ранее функций из числа перечисленных, а по завершении их работы выполнит выгрузку данных. Любая из этих функций, вызванная в момент «ожидания», завершится кодом IS_BLOCKED=8.

Функция возвращает код согласно таблице возвращаемых кодов.

Построение модели голоса диктора («базовые» функции)

API функции (источник - WAV-файл):

```
spk_model_base_ptr getModel_FromFile(int& resCode, int& modelSize,
const char* srcFilePath);
```

resCode - адрес для записи кода завершения операции согласно табл. 7.1;
modelSize - адрес для записи значения размера векторной модели;
srcFilePath - путь к WAV-файлу.

API функции (источник – буфер памяти):

```
spk_model_base_ptr getModel_FromBuf(int& resCode, int& modelSize,
const short* soundData, int soundDataSz);
```

resCode - адрес для записи кода завершения операции согласно табл. 7.1;
modelSize - адрес для записи значения размера векторной модели;
soundData - указатель на массив отсчетов;
soundDataSz - количество отсчетов в аудио буфере.

В случае успеха функция возвращает указатель на вектор-модель голоса диктора, а в *resCode* записывает код 0, иначе – возвращает nullptr, а в *resCode* пишет код ошибки.

NB: При обработке звуковых данных большого объёма велика вероятность того, что в потоке присутствует речь более, чем одного диктора. Модель голоса в таком случае будет некорректной. Поэтому рекомендуется разбивать входной поток на части длительностью не более 5 минут каждый. Такой подход к использованию функции также обеспечит оптимальный расход памяти, необходимой для построения модели голоса диктора.

Построение модели голоса диктора («расширенные» функции)

API функции (источник - WAV-файл):

```
spk_model_base_ptr getModel_FromFileExt(int& resCode, int& modelSize,
const char* srcFilePath, int& spAmount, int& sigTotalAmount);
```

resCode - адрес для записи кода завершения операции согласно табл. 7.1;
modelSize - адрес для записи значения размера векторной модели;
srcFilePath - путь к WAV-файлу;
spAmount - адрес для записи значения общей длительности речевых сегментов, на которых построена дикторская модель (мс);
sigTotalAmount - адрес для записи значения общей длительности обработанного аудио сигнала (мс).

API функции (источник – буфер памяти):

```
spk_model_base_ptr getModel_FromBufExt(int& resCode, int& modelSize,
const short* soundData, int soundDataSz, int& spAmount, int&
sigTotalAmount);
```

resCode - адрес для записи кода завершения операции;
modelSize - адрес для записи значения размера векторной модели;
soundData - указатель на массив отсчетов;

soundDataSz - количество отсчетов в аудио буфере;

spAmount - адрес для записи значения общей длительности речевых сегментов, на которых построена дикторская модель (мс);

sigTotalAmount - адрес для записи значения общей длительности обработанного аудио сигнала (мс).

Знание, на каком количестве речи построена модель, позволяет оценить её представительность (годность) для идентификации. Рекомендуемое значение *spAmount* – не менее 45000 (мс). Но допустимо использование модели и при меньшем значении. В случае успеха в *resCode* будет записан 0, иначе – код ошибки.

Построение модели голоса диктора (+VST)

API функции (источник - WAV-файл):

```
spk_model_base_ptr  getModelFromDialog_FromFileExt(int&  resCode,int& modelSize, const char* srcFilePath);
```

resCode - адрес для записи кода завершения операции;

modelSize - адрес для записи значения размера векторной модели;

srcFilePath - путь к WAV-файлу.

API функции (источник – буфер памяти):

```
spk_model_base_ptr getModelFromDialog_FromBufExt(int& resCode, int& modelSize, const short* soundData, int soundDataSz);
```

resCode - адрес для записи кода завершения операции согласно табл. 7.1;

modelSize - адрес для записи значения размера векторной модели;

soundData - указатель на массив отсчетов;

soundDataSz - количество отсчетов в аудио буфере.

Данные функции рекомендовано использовать в случаях, когда заведомо известно, что в представленной аудио сигнале два голоса. Будет выполнено разделение аудио сигнала на участки, где говорит каждый из двух дикторов, а затем построены две дикторские модели.

Возможно, что функция возвратит одну дикторскую модель. Это может быть связано либо с ошибкой сегментации, либо аудио сигнал содержит речь преимущественно одного диктора. В случае успеха в *resCode* будет записан 0, иначе - код ошибки.

Сравнение векторных моделей голосов

API функции:

```
int compareModels(float& confidence, const spk_model_base_ptr m1, const spk_model_base_ptr m2);
```

confidence - адрес для записи значения достоверности результата идентификации (сравнения двух голосов);
m1 - ссылка на 1-ю векторную модель;
m2 - ссылка на 2-ю векторную модель.

```
int compareModelsExt(float& confidence, const spk_model_base_ptr m1, const spk_model_base_ptr m2, float& distance);
```

confidence - адрес для записи значения достоверности результата идентификации (сравнения двух голосов);
m1 - ссылка на 1-ю векторную модель;
m2 - ссылка на 2-ю векторную модель;
distance - адрес для записи значения «чистого» косинусного расстояния между двумя i-векторами.

Для идентификации производится вызов данной функции, где в качестве m1 передаётся указатель на модель «целевого» диктора, а в качестве m2 – «неизвестного», или наоборот. В переменную *confidence* будет записана достоверность того, что идентифицируемый диктор является «целевым». Значение достоверности имеет диапазон [0;1]. Как видно на рисунке 1 равновероятная ошибка достигается при пороговом значении на достоверность равном 0.5. Для получения более достоверного результата рекомендуется принимать результаты, достоверность которых выше 0.5. Стоит учитывать, что в таком случае возрастает вероятность пропуска «целевого» диктора. При пороге достоверности ниже 0.5 увеличивается вероятность приёма «чужого» диктора за «целевого».

Предпочтительно, чтобы модели m1 и m2 были построены при одинаковых значениях параметра *useInternalSD*, значение которого задаётся при инициализации библиотеки.

Сравниваемые модели должны быть построены с использованием одинакового набора UBM и PCA. Функция возвращает 0 в случае успеха, иначе - код ошибки.

Очистка и удаление выходной структуры результатов

API функции:

```
void deleteData(spk_model_base_ptr dataPtr);
```

dataPtr - указатель на векторную модель голоса, которая возвращается функцией loadModel(), loadModelExt(), getModel_FromBuf(), getModel_FromFile(), getModel_FromBufExt() и getModel_FromFileExt().

```
void deleteModels(spk_models_ptr dataPtr);
```

dataPtr - указатель на список моделей голосов, который возвращается функциями getModelFromDialog_FromBufExt() и getModelFromDialog_FromFileExt().

Получение строки с текстом ошибки

API функции:

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

code - код завершения операции, возвращаемый функциями библиотеки.

Функция возвращает указатель на строку с текстовой интерпретацией кода ошибки.

Сохранение векторной модели голоса в бинарный файл

API функции:

```
int saveModel(const spk_model_base_ptr model, int modelSize, const char* dstFilePath);
```

model - ссылка на сохраняемую векторную модель;
modelSize - размер модели;
modelID - идентификатор модели;
dstFilePath - путь к файлу, в который будет сохранена модель.

```
int saveModelExt(const spk_model_base_ptr model, int modelSize, const char* dstFilePath, int spAmount, int sigTotalAmount);
```

model - ссылка на сохраняемую векторную модель;

modelSize - размер модели;

modelID - идентификатор модели;

dstFilePath - путь к файлу, в который будет сохранена модель;

spAmount - общая длительность участков речи, на которых построена модель;

sigTotalAmount - общая длительность обработанного аудио сигнала.

Функция возвращает код согласно таблице возвращаемых кодов.

Загрузка векторной модели голоса из бинарного файла

API функции:

```
spk_model_base_ptr loadModel(int& resCode, int& modelSize, const char*
srcFilePath);
```

resCode - адрес для записи завершения работы функции;

modelSize - адрес для записи размера модели;

modelID - адрес для записи идентификатора модели;

srcFilePath - путь к файлу с моделью.

```
spk_model_base_ptr loadModelExt(int& resCode, int& modelSize, const
char* srcFilePath, int& spAmount, int& sigTotalAmount);
```

resCode - адрес для записи завершения работы функции;

modelSize - адрес для записи размера модели;

modelID - адрес для записи идентификатора модели;

srcFilePath - путь к файлу с моделью;

spAmount - общая длительность участков речи, на которых построена модель;

sigTotalAmount - общая длительность обработанного аудио сигнала.

В случае успеха функция возвращает ссылку на загруженную векторную модель, а в *resCode* записывает код 0 (см. Таблицу 7.1), иначе - возвращает nullptr, а в *resCode* будет записан код ошибки.

6. ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

Требования к входным аудио данным 3i Speaker ID SDK:

- возможные источники: WAV-файлы, буфер отсчётов;
- частота дискретизации сигнала: 8 кГц;
- разрядность квантования: 8 бит, 16-бит;
- тип кодирования, если источником является WAV-файл: A-закон, Mu-закон или PCM (без дополнительных чанков типа list или fact);;
- тип кодирования, если источником является буфер памяти: PCM.

Требования к качеству сигнала:

- значение ОСШ должно составлять не менее 10дБ;
- допускается присутствие посторонних звуков в виде однотональных гудков и сигналов тонального набора, DTMF.

Выходные данные: число с плавающей точкой confidence, отражающее достоверность того, что идентифицируемый диктор является «целевым». Значение достоверности имеет диапазон [0;1].

Равновероятная ошибка между ошибками первого и второго рода достигается при пороговом значении на достоверность равном 0.5.

7. СООБЩЕНИЯ ОПЕРАТОРУ

Все сообщения оператору реализуются через коды возвращаемых значений функциями модуля, далее дается расшифровка (см. Таблицу 7.1).

Таблица 7.1 Коды возвращаемых значений функциями модуля

Код	**Текстовая интерпретация**	**Код**	**Текстовая интерпретация**
0	NO_ERR	**11**	INITIALIZATION_ERROR
1	UNKNOWN_ERROR	**12**	DEINITIALIZATION_ERROR
3	MEMORY_ALLOCATION_ERROR	**27**	INSUFFICIENT_SIZE_OF_THE_SOUND_BUFFER
4	INCORRECT_POINTER	**29**	UNABLE_TO_SAVE_FILE
5	INCORRECT_INPUT_PARAM	**30**	UNABLE_TO_OPEN_FILE
6	INCORRECT_INPUT_CONTAINER	**32**	INCORRECT_FILE_FORMAT
7	INCORRECT_INIT_PARAMS	**35**	UNABLE_TO_GET_FEATURES
8	IS_BLOCKED	**41**	UNABLE_TO_LOAD_MODEL
9	IS_NOT_INIT	**44**	UNABLE_TO_LOAD_PCA_DATA
10	ALREADY_INIT	**54**	UNABLE_TO_GET_IVECTOR

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

В настоящем документе приняты следующие условные обозначения:

СПО	Специальное программное обеспечение
СУБД	Система управления базами данных
3i Speaker ID SDK	СПО «3i Speaker ID 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
DLL	сокр. англ. Dynamic Link Library – динамическая библиотека, позволяющая многократное использование различными программными приложениями.
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.
WFST	сокр. англ. Weighted Finite State Transducer, взвешенный конечно-автоматный преобразователь, автомат Мили со взвешенными дугами.
SDK	сокр. англ. Source Development Kit – комплект средств разработки, который позволяет специалистам по программному обеспечению создавать приложения.

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.
DTMF	сокр. англ. Dual-Tone Multi-Frequency, двухтональный многочастотный аналоговый сигнал, используемый для набора телефонного номера.
UBM	сокр. англ. Universal Background Model – статистическая модель пространства признаков голоса.
VST	сокр. англ. Voice Segmentation Technology – технология сегментации по голосам
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.

Лист регистрации изменений

Изм.	Номера листов (страниц)				Всего листов (страниц) в документе	№ документа	Входящий № сопроводительного документа и дата	Подпись	Дата
	Изме-нен-ных	замененных	новых	аннулированных					