УТВЕРЖДЕН

ДССЛ.00107-08 31 01 - ЛУ

СПЕЦИАЛЬНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

«3i Speech Transcriptor»

Описание применения

ДССЛ.00107-08 31 01

Листов 62

Литера $О_1$

2016

АННОТАЦИЯ

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

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

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

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

В разделе 4 даются сведения об установке 3i Speech Transcriptor.

В разделе 5 приводится описание интерфейса программирования 3i Speech Transcriptor.

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

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

По вопросам сопровождения данного программного обеспечения можно обращаться по электронной почте: support@dss-lab.ru и телефону: +7 (495) 645 44 70. Время обращения, в зависимости от договорных условий сервисного или гарантийного обслуживания, по рабочим дням с 10:00 до 18:00 часов (время московское) или круглосуточно.

СОДЕРЖАНИЕ

1. НАЗНАЧЕНИЕ ПРОГРАММЫ

Специальное программное обеспечение «3i Speech Transcriptor» предназначено для решения задач распознавания речи. 3i Speech Transcriptor 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 dynamic linked library (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++.

2. УСЛОВИЯ ПРИМЕНЕНИЯ

2.1. Для функционирования 3i Speech Transcriptor 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 Transcriptor на вычислительной системе должно быть установлено следующее общее программное обеспечение:

- операционная система – Microsoft Windows 7 SP1 или выше либо ОС на основе ядра Linux;
- для функционирования на ОС Windows: распространяемый пакет Microsoft VC 2013 Redist (x64);

3. ОПИСАНИЕ ЗАДАЧИ

3.1. Технологии работы с речевыми данными

Возможности 3i Speech Transcriptor распознавания речи базируются на технологиях DNN и WFST – глубоких нейронных сетей (deep neural networks) и взвешенных конечных автоматов (weighted finite state transducer) для качественного дикторонезависимого распознавания слитной речи с очень большим (сотни тысяч слов) словарем, предназначенные соответственно для моделирования акустических элементов речи и моделирования лексического и лингвистического уровня речи. Модуль распознавания речи настраивается обученной заранее моделью на язык и тип канала. Указанные технологии распознавания речи обеспечивают:

- высокую скорость обработки речевого сигнала, за счет распараллеливания вычислений;
- современное качество распознавания речи (зависит от типа языка и канала);
- возможность гибкой настройки модуля распознавания речи на тип канала (телерадиовещательный - broadcast, телефония - phone) и\или язык за счет использования обучаемых моделей, распространяемых независимо от модуля;
- дикторонезависимое распознавание слитной речи, в том числе при наличии акцента, внешних шумов, неречевых звуков, музыки;
- распознавание файлов или потоков речи неограниченной длины за счет деления записей по паузам внутри речи и распознавания получившихся кусочков в отдельных потоках центрального процессора;
- большой словарь распознаваемых слов, включающий сотни тысяч слов, чего практически достаточно для распознавания любого текста общей лексики.

Набор модулей 3i Speech Transcriptor предоставляет пользователю развитой функционал работы с речевыми сигналами:

- преобразование речь-текст, русский-телефония и английский-телефония, то есть дикторонезависимое распознавание слитной речи для очень большого словаря в записи речевого сигнала и преобразование распознанного в текст.

3.2. Состав 3i Speech Transcriptor

СПО 3i Speech Transcriptor включает следующие части:

- 3i Speech Transcriptor (phone-broadcast) (русский-английский) – модуль, содержащий функциональные возможности преобразования речь-текст, русский/английский- телерадиовещательный/телефония.
- Модели речи:
  SM_EN_PH_CN_TD6W_V023 (английская телефония);
  SM_EN_TV_NS_TD6W_V017 (английский броадкаст);
  SM_RU_PH_CN_TD6W_V020 (русская телефония);
  SM_RU_TV_NS_TD6W_V026 (русский броадкаст).

4. ВЫЗОВ И ЗАГРУЗКА ПРОГРАММЫ

4.1. Установка 3i Speech Transcriptor

Установка модулей (библиотек) 3i Speech Transcriptor осуществляется переносом (копированием) архива с модулями с загрузочного диска в требуемую директорию на жестком диске сервера. После переноса (копирования) архива модулей, динамические библиотеки (SO) модулей разархивируются из него в требуемую директорию на жестком диске сервера. Модули (библиотеки) 3i Speech Transcriptor 8 готовы к эксплуатации.

4.2.Получение лицензионного файла-ключа

Бинарные файлы модулей 3i Speech Transcriptor 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 различна в зависимости от ОС целевой платформы.

Процедура получения файла-ключа для библиотеки под ОС Windows такова:

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 файл.

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.

Файл идентификационного кода системы hardware_id.hid следует передать в техническую поддержку 3i, сопровождая информацией о лицензии.

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.

4.3.Проверка работоспособности 3i Speech Transcriptor

Вы можете протестировать работоспособность 3i Speech Transcriptor путем использования одного из пакетных файлов, в зависимости от языка и канала звукового файла:

1) decode_eng_phone.bat - для английской телефонии
2) decode_eng_tv.bat - для английского броадкаста
3) decode_rus_phone.bat - для русской телефонии
4) decode_rus_tv.bat - для русского броадкаста

Входные файлы для тестирования должны быть в формате WAV PCM, 16 бит, 16 кГц для броадкаста, 8 кГц для телефонии.

Имя файла указывается в тексте пакетного файле, для этого откройте текстовым редактором выбранный вами пакетный файл и укажите полный или относительный путь к тестируемому файлу перед знаком '>', например в файле

decode_rus_tv.bat замените подстроку "./input\demo_tv_rus\d2.wav" на путь к файлу который вы используете:

```
ASR_SDK_Example SM_RU_TV_NS_TD6W_V026 /input\demo_tv_rus\d2.wav > output.txt
```

Вы можете оставить существующие указанные по умолчанию в пакетных файлах пути к тестируемым файлам, если желаете тестировать на проверочных файлах, входящих в поставку.

Проверочные файлы расположены в директории input, находящейся в директории с ASR SDK.

Убедитесь, что вы имеете лицензионный ключ, для подробностей его получения см. файл Readme.txt или п.4.2 настоящего руководства.

После указания входного файла для тестирования, запустите пакетный файл. Через некоторое время (зависит от длины файла, следует ориентироваться на количество времени равное длине файла в секундах) процесс декодирования закончится и появится файл output.txt, содержащий результат декодирования.

Результат представляет собой последовательность слов, сопровожденных временными метками (метки начала и конца данного слова относительно начала файла). Если результат присутствует, то работоспособность 3i Speech Transcriptor подтверждена.

5. ВЫПОЛНЕНИЕ ПРОГРАММЫ

5.1. Комплект модулей 3i Speech Transcriptor

Библиотека 3i Speech Transcriptor имеет классический интерфейс, характерный для динамически подгружаемых библиотек (dll или so). В комплекте с каждым модулем поставляется:

- бинарный файл динамически связываемой библиотеки (ASRDecoderLib.dll);
- бинарный файл прокси-библиотеки для статического связывания приложения пользователя с динамически связываемой библиотекой (ASRDecoderLib.lib);
- файл-хидер, содержащий исходный код интерфейса модуля на c++ для сборки связывания динамической библиотеки модуля и приложения пользователя (ASRDecoderLib.h);
- модели речи (русская телефония, русский броадкаст, английская телефония, английский броадкаст).

Описание функционала API для библиотеки 3i Speech Transcriptor представлено ниже.

5.2. Описание функционала API

Данные модули отличаются только моделями, настраивающими транскриптор на язык и тип канала, поэтому модули имеют одинаковый API. SDK предоставляет API для использования функций библиотеки, которое реализовано в файле ASRDecoderLib.h. Этот заголовочный файл содержит пространство имен asr_sdk, содержащее функции API SDK, а также коды ошибок, которые возвращают функции.

Список функций API Speech Transcriptor представлен в таблице 5.2.5.

Таблица 5.2.5 Список функций API 3i Speech Transcriptor

Наименование функции	**Описание функции**
init	Инициализация библиотеки
process_file	Функция декодирования речи из файла
process_buf	Функция декодирования речи из буфера
freeStr	Функция освобождения памяти под строку результата
deInit	Функция освобождения памяти под модели и деинициализации SDK
getErrMsg	Функция получения строкового пояснения кода ошибки SDK
init_mnt	Функция инициализации библиотеки с настройкой количества используемых ядер
AbortProcess	Функция прерывания процесса декодирования SDK
SetTopN	Функция установки количества лучших гипотез во временных срезах, попадающих в решётку гипотез
GetLattice	Функция, возвращающая решётку гипотез декодированного файла или буфера
set_model_slot	Функция установки текущего именованного слота модели
get_progress	Функция получения прогресса текущего декодирования
get_info_samplerate	Функция получения частоты дискретизации сигнала, которая соответствует текущей модели

Подробное описание функций API 3i Speech Transcriptor (phone, broadcast) (русский, английский) представлено ниже.

1. Функция инициализации библиотеки

API функции:

```
// LIBRARY INITIALIZATION
// [IN] pathToModel - path to folder with mdef,DNN,dicts,arpa
// Returns code of ending of operation
ASRLIB_API(int) init(const char* pathToModel);
```

Параметры:

pathToModel - путь к директории, содержащей модели.

Возвращает ASR_FAIL, если библиотека не инициализирована (по разным причинам).

В случае корректного завершения возвращает NO_ERR (=0) или возвращает ASR_SSE4_ENABLED (=1013), если CPU или ОС не поддерживает AVX, поэтому ASR SDK использует математические функции, оптимизированные под расширение SSE4.

Функция работает в контексте именованного слота модели, указанного последним вызовом функции *set_model_slot*. Если функция *set_model_slot* не была вызвана, работа производится со слотом по умолчанию. Все остальные слоты при этом остаются без изменений.

2. Функция декодирования речи из файла

API функции:

```
// LIBRARY PROCESS FILE
// [IN] pathToFile - full path to *.wav file needed to be decoded
// [OUT] return_code - returns code of ending of operation
// [OUT] res_strlen - length of resulting string
//Returns pointer to resulting string with contents of *.cn file
ASRLIB_API(const char*) process_file(const char* pathToFile, int&
return_code, int& res_strlen);
```

Параметры:

pathToModel – полный путь к файлу, путь не должен содержать русских символов;

return_code – код возврата, NO_ERR (=0) если функция завершилась успешно, ASR_ABORTED – если декодирование было прервано функцией AbortProcess;

res_strlen – длина строки результата;

Возвращает указатель на строку результата, выходной параметр, память для строки выделяет библиотека, строка оканчивается нулем.

Функция работает в контексте именованного слота модели, указанного последним вызовом функции *set_model_slot*. Если функция *set_model_slot* не была вызвана, работа производится со слотом по умолчанию. Все остальные слоты при этом остаются без изменений.

3. Функция декодирования речи из буфера

API функции:

```
// LIBRARY PROCESS BUF
// [IN] data - buffer of 16-bit samples
// [IN] dlen - size of buffer "data" in samples
// [OUT] return_code - returns code of ending of operation
// [OUT] res_strlen - length of resulting string
//Returns pointer to resulting string with contents of *.cn file
ASRLIB_API(const char*) process_buf(const short* data, intdlen, int&
return_code, int& res_strlen);
```

Параметры:

data – буфер отсчетов, 16 бит на отсчет;

dlen – длина буфера в отсчетах;

return_code – код возврата, NO_ERR (=0) если функция завершилась успешно, ASR_ABORTED – если декодирование было прервано функцией AbortProcess;

res_strlen – длина строки результата.

Возвращает указатель на строку результата, выходной параметр, память для строки выделяет библиотека, строка оканчивается нулем.

Функция работает в контексте именованного слота модели, указанного последним вызовом функции *set_model_slot*. Если функция *set_model_slot* не была вызвана, работа производится со слотом по умолчанию. Все остальные слоты при этом остаются без изменений.

4. Функция освобождения памяти под строку результата

API функции:

```
// LIBRARY FREE RESULT STRING
// call this function every time after calling process_file or
process_buf
// note: results of recognition returned by process_file or
process_buf can be used before freeing
// [IN] str - string to free, must be one of result strings (returned
by process_file or process_buf)
//Returns code of ending of operation
ASRLIB_API(int) freeStr(char* str);
```

Параметры:

str – указатель на строку, память под которую должна быть освобождена. Такие строки возвращаются функциями *process_file* или *process_buf*.

Функция возвращает код возврата, NO_ERR (=0) если функция завершилась успешно. Возвращает ASR_FAIL если предпринята попытка освободить память дважды либо указатель не является указателем на строку, возвращенную библиотекой.

Функция *freeStr* освобождает выделенную под строки память в любом порядке в любое время. Необходимо использовать полученные строки до освобождения памяти под них. Если память под строки не была освобождена, то вся выделенная память будет освобождена при деинициализации библиотеки.

Функция работает в контексте именованного слота модели, указанного последним вызовом функции *set_model_slot*. Если функция *set_model_slot* не была вызвана, работа производится со слотом по умолчанию. Все остальные слоты при этом остаются без изменений.

5. Функция освобождения памяти под модели и деинициализации SDK

API функции:

```
// LIBRARY DEINITIALIZATION
ASRLIB_API(void) deInit();
```

Вся выделенная и не освобожденная память под строки результата при деинициализации освобождается. Функция работает в контексте именованного слота модели, указанного последним вызовом функции *set_model_slot*. Если функция *set_model_slot* не была вызвана, работа производится со слотом по умолчанию. Все остальные слоты при этом остаются без изменений.

6. Функция получения строкового пояснения кода ошибки

API функции:

```
// GET ERR-STRING BY CODE OF ERROR
// [IN] errCode - code that returned by other library functions
// Return err-string
ASRLIB_API(const char*) getErrMsg(interrCode);
```

Параметры:

errCode – значение кода ошибки, для расшифровки.

Функция возвращает указатель на строку (оканчивающуюся 0), строковой идентификатор ошибки из таблицы возвращаемых кодов.

Функция работает в контексте именованного слота модели, указанного последним вызовом функции *set_model_slot*. Если функция *set_model_slot* не была вызвана, работа производится со слотом по умолчанию. Все остальные слоты при этом остаются без изменений.

7. Функция инициализации библиотеки с настройкой количества используемых ядер

API функции:

```
// LIBRARY INITIALIZATION
// [IN] pathToModel - path to folder with mdef,DNN,dicts,arpa
// [IN] nt - max number of threads for lib, -1 for all possible
threads
// Returns code of ending of operation
ASRLIB_API(int) init_mnt(const char* pathToModel, intmnt);
```

Параметры:

pathToModel – путь к директории, содержащей модели;

nt – количество ядер, используемых для вычислений. Значение -1 устанавливает количество ядер, равное всем доступным ядрам на данной системе.

Возвращает ASR_FAIL, если библиотека не инициализирована (по разным причинам). Возвращает ASR_WRONG_NUM_OF_THREADS, если количество ядер задано некорректно.

В случае корректного завершения возвращает NO_ERR (=0) или возвращает ASR_SSE4_ENABLED (=1013), если CPU или ОС не поддерживает AVX, поэтому ASR SDK использует математические функции, оптимизированные под расширение SSE4.

Функция работает в контексте именованного слота модели, указанного последним вызовом функции *set_model_slot*. Если функция *set_model_slot* не была вызвана, работа производится со слотом по умолчанию. Все остальные слоты при этом остаются без изменений.

8. Функция прерывания процесса декодирования SDK

API функции:

```
// LIBRARY ABORT CURRENT PROCESSING
// Asynchronous abort of pending decoding process
// Returns code of ending of operation
ASRLIB_API(int) AbortProcess();
```

Вызывает прекращение процесса декодирования. Процесс декодирования прекращается после обработки текущего куска сигнала, это занимает время, максимум 30 сек.

Данную функцию следует вызывать в другом потоке, отличном от потока, занятом декодированием.

Возвращает NO_ERR(=0) в случае корректного завершения, возвращает ASR_LIB_NOT_INITED, если библиотека не инициализирована.

После прерывания декодирования возможно продолжить рабочий цикл библиотеки, то есть вызывать другие функции декодирования\инициализации.

Функция работает в контексте именованного слота модели, указанного последним вызовом функции *set_model_slot*. Если функция *set_model_slot* не была вызвана, работа производится со слотом по умолчанию. Все остальные слоты при этом остаются без изменений.

9. Функция установки количества лучших гипотез во временных срезах, попадающих в решётку гипотез

API функции:

```
// LIBRARY SET TOP N HYPOTHESES IN LATTICE
// [IN] newtopn - quantity of max value of hypotheses in lattice
// Returns code of ending of operation
ASRLIB_API(int) SetTopN(intnewtopn);
```

Параметры:

newtopn – устанавливаемое максимальное количество лучших гипотез в любом временном срезе, попадающих в решетку гипотез, должно быть больше 0 и меньше 100.

Возвращает NO_ERR(=0) в случае корректного завершения, возвращает ASR_LIB_NOT_INITED, если библиотека не инициализирована, возвращает ASR_WRONG_NUM_OF_HYPS, если количество гипотез некорректно (меньше 1 или больше 100).

Данную функцию следует использовать перед декодированием файла или буфера. Установленное функцией значение сохраняется до деинициализации библиотеки.

Функция работает в контексте именованного слота модели, указанного последним вызовом функции *set_model_slot*. Если функция *set_model_slot* не была вызвана, работа производится со слотом по умолчанию. Все остальные слоты при этом остаются без изменений.

10. Функция, возвращающая решётку гипотез декодированного файла или буфера

API функции:

```
// LIBRARY GET LATTICE
// [OUT] return_code - returns code of ending of operation
// [OUT] res_strlen - length of resulting string
// Returns pointer to resulting string with lattice
ASRLIB_API(const char*) GetLattice(int& return_code, int& res_strlen);
```

Параметры:

return_code – код возврата, NO_ERR (=0) если функция завершилась успешно, ASR_ABORTED – если декодирование было прервано функцией AbortProcess, ASR_LIB_NOT_INITED, если библиотека не инициализирована;

res_strlen – длина строки результата.

Возвращает указатель на строку решётки гипотез, выходной параметр, память для строки выделяет библиотека, строка оканчивается нулем. Данную функцию следует использовать после декодирования файла или буфера. Строку решётки гипотез следует использовать до декодирования следующего файла. Нет необходимости освобождать память под строку результата, память будет освобождена во время декодирования следующего результата либо во время деинициализации библиотеки.

Функция работает в контексте именованного слота модели, указанного последним вызовом функции *set_model_slot*. Если функция *set_model_slot* не была

вызвана, работа производится со слотом по умолчанию. Все остальные слоты при этом остаются без изменений.

11. Функция установки текущего именованного слота модели

API функции:

```
// LIBRARY SET MODEL NAMED SLOT
// [IN] slot_name - identifier string of named slot. If such a slot
isn't exist, it will be created
// Returns 1 if named slot exist, 0 if new named slot created
ASRLIB_API(int) set_model_slot(const char* slot_name);
```

Параметры:

slot_name – имя слота модели, однозначно идентифицирует устанавливаемый слот. Если слот с таким именем не существует, то он будет создан. Возвращает 1 если указанный в slot_name именованный слот существует, 0 если слот не существовал и был создан.

Именованный слот модели задает контекст, в котором работают все остальные функции модуля, то есть каждый слот ассоциирован с загруженной в него моделью, эту модель можно использовать для распознавания, установив данный слот.

Функции модуля, работающие в контексте одного слота, не влияют на состояние и содержимое других слотов. Если перед использованием любой функции модуля (кроме самой *set_model_slot*) не была вызвана функция *set_model_slot*, то используется именованный слот по умолчанию (с пустым именем). Если клиентское приложение не планирует использовать несколько моделей, загрузив их все заранее в память, то в использовании функции *set_model_slot* нет необходимости. Таким образом, достигается совместимость модуля с кодом клиентов, написанных для более ранних версий модуля (при этом обязательна пересборка этого кода с использованием нового хидера модуля, поскольку меняются порядковые номера функций в интерфейсе библиотеки модуля).

Общий способ использования данной функции:

1) вызвать функцию *set_model_slot* с параметром А, создав именованный слот А

2) вызвать функцию *init* или *init_mnt*, загрузив модель ассоциированную со слотом А

3) вызвать функцию *process_buf*, использовав загруженную и ассоциированную со слотом А модель для декодирования буфера (возможно, неоднократно)

4) вызвать функцию *set_model_slot* с параметром Б, создав именованный слот Б

5) вызвать функцию *init* или *init_mnt*, загрузив модель ассоциированную со слотом Б

6) вызвать функцию *process_buf*, использовав загруженную и ассоциированную со слотом Б модель для декодирования буфера (возможно, неоднократно)

7) вызвать функцию *set_model_slot* с параметром А, установив именованный слот А

8) вызвать функцию *process_buf*, использовав загруженную и ассоциированную со слотом А модель для декодирования буфера (возможно, неоднократно)

9..n-1) ...

n) вызвать функцию *set_model_slot* с параметром А, установив именованный слот А

n+1) вызывать функцию *deInit*, выгрузив модель ассоциированную со слотом А

n+2) вызвать функцию *set_model_slot* с параметром Б, установив именованный слот Б

n+3) вызывать функцию *deInit*, выгрузив модель ассоциированную со слотом Б

Обратите внимание, функция работает синхронно, попытка вызвать ее из другого потока приведет к ожиданию вызова в очереди. Функция *set_model_slot* предназначена для попеременного, синхронного использования загруженных моделей с целью упростить загрузку и выгрузку моделей, которая будет осуществляться один раз за время работы клиентского приложения.

12. Функция получения прогресса текущего декодирования

API функции:

```
// LIBRARY GET PROGRESS OF DECODING
// Returns progress in percent
ASRLIB_API(int) get_progress();
```

Возвращает прогресс декодирования в процентах, целое число в диапазоне 0..100. Возвращает число вне этого диапазона, будучи вызвана в момент, когда декодирование не производится.

Данную функцию следует вызывать в другом потоке, отличном от потока, занятом декодированием.

13. Функция получения частоты дискретизации сигнала, которая соответствует текущей модели

API функции:

```
// LIBRARY GET MODEL SAMPLE RATE
// Returns sample rate of model, 0 if no initialized model
ASRLIB_API(int) get_info_samplerate();
```

Возвращает частоту дискретизации сигнала в Гц, под которую была обучена текущая модель. Возвращает 0, если модель в текущем слоте не инициализирована.

6. ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

6.1. Модули 3i Speech Transcriptor (phone, broadcast) (русский, английский)

Данные модули отличаются только моделями, настраивающими транскриптор на язык и тип канала, поэтому модули имеют одинаковые входные и выходные данные.

Входные данные:

- файл WAV PCM, моно, 16 бит, частота дискретизации зависит от настроек акустической модели;
- буфер отсчетов, моно, 16 бит, частота дискретизации зависит от настроек акустической модели.

Выходные данные: строка, заканчивающаяся 0, содержащая в себе декодированный текст, слова разделены построчно (разделитель \n, символ 10), сопровождены информацией о порядковом номере слова, времени начала и времени конца, а также зарезервированным значением оценки уверенности.

Пример строки слова: *T=2 ST=1.21 ET=1.62 W=ЕСЛИ P=1*

T - порядковый номер слова,

ST - время начала слова (в секундах от начала файла или буфера),

ET - время окончания слова,

W - слово (кодировка UTF8),

P - зарезервированное значение оценки уверенности распознавания.

Также возможно получить дополнительно к строке с декодированным текстом решётку гипотез, которая представляет собой строку, содержащую гипотезы сегментов (слов и филлеров), сегменты разделены построчно (разделитель \n, символ 10), сопровождены информацией о времени начала и времени конца, об акустической и полной уверенности распознавания.

Пример строки сегмента: *ST=3310 ET=3850 W=МЕТРО AS=57.223 LS=42.8072*

ST - время начала слова (в мс от начала файла или буфера),

ET - время окончания слова,

W - слово (кодировка UTF8),

AS - акустическая уверенность распознавания (чем больше, тем больше надежность распознавания без привлечения контекстной языковой информации),

LS - полная уверенность распознавания (чем больше, тем больше надежность распознавания данного сегмента в целом).

7. СООБЩЕНИЯ ОПЕРАТОРУ

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

7.1. Модули 3i Speech Transcriptor (phone, broadcast) (русский, английский)

Таблица 7.4.1 Список возвращаемых кодов

Код	**Значение**
1	NO_ERR=0, операция выполнена корректно;
2	ASR_FAIL=1000, операция выполнена некорректно, причина не дифференцирована;
3	ASR_WRONGFILEPATH=1001, неверный путь к файлу;
4	ASR_FAIL_TO_CREATE_RES=1002, зарезервированный код ошибки;
5	ASR_LIB_NOT_INITED=1003, попытка вызова функции модуля при не инициализированной модели
6	ASR_BUF_EMPTY=1004, буфер отсчетов для декодирования пуст;
7	ASR_WRONG_NUM_OF_THREADS=1005, количество ядер задано неверно;
8	ASR_ABORTED=1006, операция идентификации была прервана, строка результата некорректна
9	ASR_WRONG_NUM_OF_HYPS=1007, задано неверное количество лучших гипотез во временной решётке гипотез, количество меньше 1 или больше 100;
10	ASR_FAIL_TOREAD_WAV=1008, нельзя прочитать отсчёты из файла с речью;
11	ASR_FAIL_TOOPEN_WAV=1009, нельзя открыть файл с речью;

Код	Значение
12	ASR_SSE4_ENABLED=1013, включен режим использования расширения процессора SSE4 вместо AVX

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

В настоящем документе приняты следующие условные обозначения:

СПО	Специальное программное обеспечение
3i Speech Transcriptor	СПО «Speech Transcriptor» с указанием обрабатываемых данных
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.

Лист регистрации изменений									
Изм.	Номера листов (страниц)				Всего листов (страниц) в документе	№ документа	Входящий № сопроводительного документа и дата	Подпись	Дата
	Изменен-ных	замененных	новых	аннулированных					