УТВЕРЖДЕН

ДССЛ.00105 - ЛУ

СПЕЦИАЛЬНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

«3i Language ID»

Описание применения

ДССЛ.00105

Листов 24

Литера $О_1$

2016

АННОТАЦИЯ

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

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

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

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

В разделе 4 даются сведения об установке 3i Language ID.

В разделе 5 приводится описание интерфейса программирования 3i Language ID.

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

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

По всем вопросам, связанным с использованием СПО «3i Language ID» 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

1. НАЗНАЧЕНИЕ ПРОГРАММЫ

Специальное программное обеспечение «3i Language ID» предназначено для решения задачи идентификации языка по голосу. 3i Language ID 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 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 Language ID 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 Language ID на вычислительной системе должно быть установлено следующее общее программное обеспечение:

- операционная система x64 – на основе ядра Linux;
- пакет libtbb2.

3. ОПИСАНИЕ ЗАДАЧИ

3.1 Задача идентификации языка по речевым данным

Задача идентификации является задачей классификации с открытым множеством классов. Таким образом, задача идентификации языка по речевым данным заключается в определении вероятностей принадлежности данного речевого сигнала к каждому из целевых языков. Наибольшая вероятность определяет язык, который с наибольшей правдоподобностью содержится в речевом отрезке. При этом, существует порог этой вероятности, задаваемый пользователем, превышение которого дает основание считать, что данный речевой сигнал является речью на языке, вероятность которого наибольшая среди целевых. Если же наибольшая среди целевых языков вероятность не превышает порог, то принимается решение о том, что данный речевой сигнал является речью на каком-то языке, не входящим в множество целевых.

Специфика задачи требует, чтобы анализируемый речевой сигнал являлся достаточно длинным (не менее 10 сек), чтобы учесть статистику распределения звуков и слов и содержал, в основном, только речь, без посторонних звуков, пауз, музыки. Качество речи должно быть высоким. Недопустимо идентифицировать язык по речевому сигналу, содержащему реализации речи на нескольких языках, т.е. требуется предварительное разбиение длинных записей на отрезки и определение для каждого отрезка его языка. Не исключено, что язык речевого сигнала менялся.

Метод идентификации языка

Используемая в 3i Language ID технология идентификации языка основана на вычислении наиболее вероятной последовательности звуков и слов, и определении вероятности порождения моделью целевого языка этой наиболее вероятной последовательности. Данный процесс повторяется для каждого целевого языка. Таким образом, учитывается акустическая и лингвистическая статистика распределения единиц языка (фонем\слогов\слов) на основе Скрытой

Марковской модели для моделирования последовательностей лингвистических единиц речи и глубокой нейронной сети для моделирования акустики речи. Предполагается, что речь на нецелевом языке будет иметь меньшую вероятность порождения моделью целевого языка, чем речь на целевом языке, поскольку фонетический состав, акустические особенности и распределение слов в нецелевом языке значительно отличается от целевого. Далее, используется несколько моделей целевого языка, для каждой из них находится вероятность порождения моделью данного речевого сигнала, а затем выбирается целевой язык, на модели которого получена наибольшая вероятность. При этом возможно, что истинный язык данного речевого сигнала не входит в перечень целевых языков, поэтому недопустимо интерпретировать выход процесса как идентифицированный язык без учета вероятности принадлежности идентифицированного языка целевому языку. На выходе модуля пользователь получает распределение вероятностей принадлежности обрабатываемой речевой записи данным целевым языкам, а также максимальную среди этих вероятностей. Затем пользователь принимает по этой вероятности решение, является ли язык обрабатываемой речевой записи идентифицированным языком или же языком вне множества целевых языков. Данное решение принимается пользователем по определяемому им самостоятельно порогу. Порог пользователь определяет самостоятельно на основании соотношений вероятностей ошибок первого и второго рода. Модуль 3i Language ID не в состоянии автоматически установить этот порог, поскольку неизвестно желаемое пользователем соотношение ошибок, однако предоставляет необходимую информацию для определения порога.

Каждый язык из целевого множества языков должен иметь модель языка, которая поставляется совместно с модулем. Кроме того, каждый тип канала (броадкаст, телефония) для каждого языка также имеет модель языка.

Обрабатываемую запись рекомендуется очистить от пауз, неречевых звуков, музыки, для этого рекомендуется воспользоваться модулем 3i Speech Detector SDK.

4. ВЫЗОВ И ЗАГРУЗКА ПРОГРАММЫ

4.1 Установка 3i Language ID

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

4.2 Получение лицензионного файла-ключа

Бинарные файлы модулей 3i Language ID 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) Передаётся шаблон ключа в формате *. Grdvd, серийный номер в текстовом файле, пакет linux-sp_7.0.tar.gz.
2) Устанавливаем Curl на рабочей станции пользователя при помощи команды:
   apt-get install curl
3) Распаковываем пакет linux-sp_7.0.tar.gz:
   tar -xvf linux-sp_7.0.tar.gz -C /<путь для распаковки>
4) Из распакованного пакета запускаем демон:
   sudo ./grddaemon

5) Из распакованного пакета запускаем утилиту активации в режиме оффлайн. При этом необходимо указывать путь к файлу с серийным номером. В этом примере все файлы находятся в одной директории:

   ./GrdSPActivate ./SPTestUbuntu.grdvd /serialfile=./sn /offline

   здесь:

   ./GrdSPActivate утилита активации

   ./SPTestUbuntu.grdvd шаблон софтверного ключа

   /serialfile=./sn параметр, определяющий путь к файлу с серийным номером (sn - имя файла)

   /offline параметр указывающий на тип активации, оффлайн

6) Утилита сообщит о том, что создан файл, одноименный с Вашим шаблоном, и имеющий расширение *.grdvd.toserver
7) Данный файл передаём поставщику ПО(ООО “ДСС Лаб”) для активации.
8) После получения файла *.grdvd.fromserver запускаем утилиту активации из терминала, и указываем ей в параметрах путь к файлу *.grdvd.fromserver:

   ./GrdSPActivate SPTestUbuntu.grdvd.fromserver

9) утилита сообщает, что активация выполнена успешно.

5. ВЫПОЛНЕНИЕ ПРОГРАММЫ

5.1 Комплект модуля 3i Language ID

Модуль 3i Language ID имеет 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, каждая из которых содержит следующие файлы:

1. mdef - акустическая модель;
2. fillers - словарь обозначений неречевых звуков;
3. transition_matrices - матрицы переходных вероятностей для HMM, акустическая модель;
4. wfst_carcs - структура автомата, описывающая фоны;
5. wfst_cl - структура автомата, описывающего лексику;
6. wfst_gt - структура автомата, описывающего грамматику (языковую модель);
7. wfst_words - выходные символы автомата (слова);
8. dsp.conf - настройки акустической модели;
9. matrix.lda - матрица для линейного дискриминантного анализа, акустическая модель (этот файл присутствует, если ключ lda_dim в dsp.conf больше 0);
10. dnn.net - структура и веса глубокой нейронной сети (DNN), акустическая модель;
11. dnn.prob - априорные вероятности фонов, акустическая модель;

Не рекомендуется помещать файлы из директории моделей в директорию, содержащую 3i Language ID, поскольку версии моделей независимы от версии 3i Language ID, это приведет к сложности обновления и переключения моделей и путанице. Рекомендуется хранить директорию моделей отдельно от директории 3i

Language ID. Каждый тип канала (и язык) имеет собственную директорию моделей и рекомендуется хранить ее отдельно от директорий моделей других типов канала. Не следует смешивать файлы из директорий моделей для различных типов канала (и разных версий моделей одного типа канала).

− директория lang_id_example, содержащая проект примера использования библиотеки на c++;

− lang_id_test тестовое приложение - пример использования библиотеки на c++.

5.2 Процесс функционирования модуля

1. Инициализация моделей: клиентское приложение вызывает функцию установки текущего слота и инициализации библиотеки для каждого целевого языка. Установка используемого именованного слота модели: клиентское приложение функцией set_model_slot устанавливает текущий именованный слот модели. Если данный слот не существовал, то он будет создан. Все функции после вызова функции set_model_slot работают в контексте установленного именованного слота модели. Таким образом возможно содержать в ОЗУ несколько моделей, что позволит избежать ожидания загрузки и выгрузки моделей. Если функция set_model_slot не вызывается ни разу, то работа ведется в контексте именованного слота по умолчанию. Повторный вызов set_model_slot устанавливает текущим другой слот, если указано другое имя. После установки текущего слота вызывается функция инициализации модели init или init_mnt (используя эту функцию при инициализации библиотеки возможно задать количество используемых ядер). Библиотека получает путь к директории, содержащей файлы моделей, загружает эти данные в память, инициализирует структуры моделей. Данный процесс занимает несколько секунд. Модели занимают значительный объём ОЗУ (до 1 Гб и более, в зависимости от модели). Таким образом, для использования библиотеки следует инициализировать модели для каждого целевого языка и каждому целевому языку назначить именованный слот. В дальнейшем, имя слота для целевого языка является его идентификатором.

2. Рабочий цикл: клиентское приложение вызывает функцию идентификации либо файла либо буфера с отсчетами, происходит процесс идентификации, результат (идентификатор целевого языка и вероятность порождения обработанной речи моделью целевого языка) возвращается клиентскому приложению. Также клиентское приложение может запросить распределение вероятностей порождения по моделям всех целевых языков.

Возможно любой момент прервать процесс декодирования, а также запросить из отдельного потока прогресс декодирования в процентах.

В рабочем цикле клиентское приложение может вызывать функцию идентификации сколько угодно раз (но синхронно, в одном потоке).

3. Выгрузка моделей: клиентское приложение вызывает функцию деинициализации deInit, библиотека выгружает модели и освобождает всю занятую моделями память. Затем клиентское приложение может инициализировать другие модели целевых языков для использования в новом рабочем цикле. Также клиентское приложение может завершить работу библиотеки.

Рекомендуется использовать библиотеку в однопоточном режиме. Попытка вызова функций библиотеки из разных потоков приведет к ожиданию вызовов в очереди (это не касается функции AbortProcess и функции get_progress). Ядра загружаются максимально возможным образом (на этапе акустической обработки близко к 100%, на этапе применения модели языка близко к 70%, данные значения приведены для конкретных условий разработчика, неполная загрузка обусловлена необходимостью интенсивного обмена с ОЗУ и нехваткой кэша). В связи с этим, нет смысла использовать библиотеку в многопоточном режиме (или загружать несколько экземпляров so), поскольку распараллеливание вычислений обеспечивается библиотекой.

5.3 Описание функционала API

Список функций API 3i Language ID представлен в таблице 5.1.

Таблица 5.1 Список функций API 3i Language ID

Наименование функции	**Описание функции**
init	Инициализация библиотеки
init_mnt	Инициализация библиотеки с установкой количества используемых библиотекой вычислительных ядер
deInit	Деинициализация библиотеки
process_langid_file *process_langid_buf*	Идентификация языка речевого сигнала и вычисление вероятности порождения его моделью идентифицированного языка. ***Источник:*** буфер памяти или WAV-файл
getErrMsg	Функция получения строкового пояснения кода ошибки.
AbortProcess	Функция прерывания процесса идентификации.
get_probs	Получение вероятностей порождения последнего обработанного речевого сигнала всеми целевыми моделями.
set_model_slot	Установка текущего именованного слота модели. В дальнейшем имя слота используется как идентификатор целевого языка.
get_progress	Получение прогресса текущей идентификации.
get_info_samplerate	Функция получения частоты дискретизации сигнала, которая соответствует текущей модели.

Подробное описание функций API 3i Language ID представлено ниже.

Инициализация библиотеки

Прежде, чем приступить к работе с библиотекой, необходимо ее инициализировать. В процессе инициализации производится загрузка моделей целевых языков и им ставится в соответствие идентификатор – имя слота. Идентификатор в дальнейшем используется при возвращении результата идентификации языка.

API функций:

Функция инициализации библиотеки init:

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

Параметры:

pathToModel – путь к директории, содержащей модель

Возвращает ASR_FAIL(=1000), 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 3i Language ID 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 (с пустым именем). Не рекомендуется для целей идентификации языка использовать слот по умолчанию.

Все остальные слоты при этом остаются без изменений.

Функция инициализации библиотеки с настройкой количества используемых ядер init_mnt:

```
// LIBRARY INITIALIZATION
// [IN] pathToModel - path to folder with mdef,DNN,dicts,wfsts
```

```
// [IN] nt - max number of threads for lib, -1 for all possible threads
// Returns code of ending of operation
LANGIDLIB_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 3i Language ID 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. Не рекомендуется для целей идентификации языка использовать слот по умолчанию.

Все остальные слоты при этом остаются без изменений.

Функция установки текущего именованного слота модели set_model_slot:

```
// 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
LANGIDLIB_API(int) set_model_slot(const char* slot_name);
```

Параметры:

slot_name – имя слота модели, однозначно идентифицирует устанавливаемый слот и целевую модель. 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 идентификатор целевого языка, то есть каждый слот ассоциирован с загруженной в него моделью языка.

Обратите внимание, функция работает синхронно, попытка вызвать ее из другого потока приведет к ожиданию вызова в очереди.

Деинициализация библиотеки

По окончании работы с библиотекой необходимо выгрузить модели целевых языков, загруженных при инициализации библиотеки.

Функция освобождения памяти под модели и деинициализации SDK deInit:

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

После вызова данной функции будет заблокирована возможность идентификации.

Идентификация языка речевого сигнала

Рабочий цикл использования модуля 3i Language ID заключается в вызове функций process_langid_file и process_langid_buf для идентификации языка файла с речью и буфера с речью соответственно.

Функция идентификация языка речи в файле process_langid_file:

```
// 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
// [OUT] return_prob - return score
//Returns pointer to resulting ID string
```

```
LANGIDLIB_API(const char*) process_langid_file(const char*
pathToFile, int& return_code, int& res_strlen, float&
return_prob);
```

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 идентификация была прервана функцией
AbortProcess;
res_strlen - длина строки результата;
return_prob – оценка вероятности порождения речевого сигнала
моделью идентифицированного языка.
```

Возвращает указатель на строку результата (UTF8), выходной параметр, память для строки выделяет библиотека, строка оканчивается нулем, память строки освобождается библиотекой при деинициализации.

Функция идентификации языка речи в буфере process_langid_buf:

```
// 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 ID string
// [OUT] return_prob - return score
LANGIDLIB_API(const char*) process_buf(const short* data,
intdlen, int& return_code, int& res_strlen, float& return_prob);
```

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;
return_prob – оценка вероятности порождения речевого сигнала
моделью идентифицированного языка.
```

Возвращает указатель на строку результата (UTF8), выходной параметр, память для строки выделяет библиотека, строка оканчивается нулем, память строки освобождается библиотекой при деинициализации.

Получение строки с текстом ошибки

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

Параметры:

errCode – значение кода ошибки, для расшифровки.

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

Функция прерывания процесса идентификации AbortProcess:

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

Вызывает прекращение процесса идентификации. Процесс декодирования прекращается после обработки текущего куска сигнала, это занимает время, максимум 5 сек.

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

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

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

Функция получения прогресса текущей идентификации get_progress:

```
// LIBRARY GET PROGRESS OF DECODING
// Returns progress in percent
```

```
LANGIDLIB_API(int) get_progress();
```

Возвращает прогресс идентификации 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.

Функция получения частоты дискретизации сигнала, которая соответствует текущей модели get_info_samplerate:

```
// LIBRARY GET MODEL SAMPLE RATE
// Returns sample rate of model, 0 if no initialized model
LANGIDLIB_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

Требования к входным аудио данным 3i Language ID:

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

Частота дискретизации данных на входе должна соответствовать указанной в dsp.conf (не редактируйте этот файл) в ключе rate.

Требования к качеству сигнала:

- значение ОСШ должно составлять не менее 10 дБ;

Выходные данные: число с плавающей точкой return_prob, отражающее достоверность того, что идентифицируемый язык в речевом сигнале является «целевым». Значение достоверности имеет диапазон [0;1]. Также возвращается идентификатор целевого языка (имя слота) в виде строки, а также ее длина.

7. СООБЩЕНИЯ ОПЕРАТОРУ

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

Возвращаемые коды

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 идентификации пуст;

7. ASR_WRONG_NUM_OF_THREADS=1005, количество ядер задано неверно;

8. ASR_ABORTED=1006, операция идентификации была прервана, строка результата некорректна.

10. ASR_FAIL_TOREAD_WAV=1008, нельзя прочитать отсчёты из файла с речью;

11. ASR_FAIL_TOOPEN_WAV=1009, нельзя открыть файл с речью;

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 Language ID	СПО «3i Language ID »
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
SO	сокр. англ. Shared object – динамическая библиотека в ОС на базе ядра Linux, 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.

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	№ доку мента	Входящий № сопроводи тельного документа и дата	Под пись	Дата
	Изме-нен-ных	заменен ных	новых	аннули рован ных					