УТВЕРЖДЕН

RU.СДРТ.07.04.001-01 99 01-ЛУ

ПРОГРАММНЫЙ КОМПЛЕКС IVA TERRA

Руководство по установке

RU.СДРТ.07.04.001-01 99 01

Листов 30

2024

АННОТАЦИЯ

Настоящий документ является руководством по установке Программного комплекса «IVA Terra» RU.СДРТ.07.04.001-01 (далее – ПК IVA Terra, программный комплекс).

В документе приведены общие сведения об установке и настройке основных компонентов, входящих в состав программного комплекса, а также дополнительного программного обеспечения, необходимого для работы ПК IVA Terra.

Документ предназначен для лиц, изучающих, сопровождающих и занимающихся дальнейшей разработкой, установкой и эксплуатацией программного комплекса.

СОДЕРЖАНИЕ

1 ОБЩИЕ СВЕДЕНИЯ

1.1 Обозначение и наименование программы

Наименование программы – Программный комплекс «IVA Terra».

Обозначение программы – RU.СДРТ.07.04.001-01.

1.2 Программная среда исполнения ПК IVA Terra

Для функционирования ПК IVA Terra необходимо следующее программное обеспечение (далее – ПО):

1) система контейнеризации Docker,

2) плагин Docker Compose;

3) nvidia runtime;

4) cuda библиотеки.

Программное обеспечение IVA Terra поставляется в виде docker-образов в формате *.gz, в который входят все необходимые модули и компоненты.

Во время установки программного комплекса происходит установка следующих модулей:

– RabbitMQ;

– PostgreSQL;

– Terra API;

– Terra Offline Worker;

– Terra Online Worker;

– Terra ADP Client.

2 ИСПОЛЬЗУЕМЫЕ ТЕХНИЧЕСКИЕ СРЕДСТВА

Минимальные технические характеристики, предъявляемые к серверному оборудованию, предназначенному для функционирования программного комплекса:

– процессор Intel(R) Core(TM) i7-14700K (20 ядер с частотой 2,5 - 3,4 ГГц) или Xeon(R) Gold 6330 (28 ядер с частотой 2,0 ГГц);

– графический процессор: NVIDIA RTX 4090 или NVIDIA A16;

– оперативная память: не менее 64 ГБ;

– объём дискового хранилища (SSD): не менее 1 ТБ;

– сетевой интерфейс: не менее 1000 Мбит/с.

Серверная платформа функционирует на базе операционных систем (ОС) Ubuntu/Astra Linux и имеет поддержку систем контейнеризации.

Исполнение моделей нейронных сетей выполнятется на графических процессорах (GPU), с альтернативной возможностью выполнения на центральном процессоре сервера (CPU).

Для работы с web-интерфейсом ПК IVA Terra используется стационарный (переносной) компьютер на базе операционных систем Windows, MacOS, Linux с поддержкой браузеров.

3 ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ

3.1 Назначение программы

ПК IVA Terra предназначен для упрощения и облегчения корпоративных коммуникаций и работе с текстом, в том числе:

– перевод аудиопотоков в текст по итогам встречи (транскрибация);

– создание саммари (создание коротких записей или аннотаций);

– создание протоколов (формирование списков поручений).

3.2 Выполняемые функции

Конечным пользователям доступны следующие функции:

– транскрибатор;

– суммаризатор;

– протоколирование.

IVA Terra выполняет определение и расстановку пунктуации, чисел, дат, номеров документов.

При оффлайн-транскрибации конференции обеспечивается распознавание нескольких одновременно говорящих спикеров с указанием в текстовом документе имени спикера и времени произнесения речи.

4 ПОДГОТОВКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ УСТАНОВКИ ПК

4.1 Общие положения

Программное обеспечение ПК IVA Terra поставляется в виде docker-архива формата *.gz, в который входят все необходимые модули и компоненты.

Перед началом установки необходимо поместить установочные файлы с программным обеспечением ПК IVA Terra на жёсткий диск сервера в папку, определяемую заказчиком.

Вся работа по установке, настройке и проверке программы проводится пользователем, обладающим root-правами администратора.

Установка, настройка и проверка ПК IVA Terra осуществляется стандартными средствами и командами через командную строку (консоль) ОС сервера.

Закрытие командной строки в процессе установки может привести к нарушениям при установке и последующей переустановке программных компонентов, входящего в состав ПК IVA Terra.

4.2 Связи с другими программами

Для установки и функционирования ПК IVA Terra необходимо использовать последние актуальные версии следующего программного обеспечения:

1) программная платформа Docker (далее – Docker),

2) плагин Docker Compose (далее – Docker Compose).

Плагин Docker Compose устанавливается в процессе установки Docker.

5 ВЫЗОВ И ЗАГРУЗКА

5.1 Подготовка к установке программного комплекса IVA Terra

Для подготовки к установке ПК IVA Terra необходимо выполнить перечисленную ниже последовательность действий:

1) на сервере установить ОС Ubuntu не ниже версии 22.04.

Примечание 1. В настройках BIOS обязательно должна быть включена «проверка подписи драйверов».

Примечание 2. В процессе установки ОС, Docker устанавливать не нужно, иначе в ОС может оказаться несколько копий Docker и `--runtime=nvidia` не будет обнаруживать `nvidia-container-runtime`.

2) установить драйвера Nvidia выполнив следующие команды:

Примечание. Версия драйвера Nvidia должна быть выше 525.XX.

```
sudo ubuntu-drivers autoinstall
sudo ubuntu-drivers install --gpgpu nvidia:525-server
sudo apt install nvidia-utils-525-server

# установка 535 если 525 недоступны
sudo apt-get purge nvidia-*
sudo apt-get purge libnvidia-*
sudo apt-get purge cuda-*

sudo apt-get update
sudo apt-get autoremove

apt search nvidia

sudo apt install libnvidia-common-535
sudo apt install libnvidia-gl-535
sudo apt install nvidia-driver-535
```

3) установить Docker с Nvidia выполнив следующие команды:

```
sudo apt install apt-transport-https ca-certificates curl software-properties-common

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/trusted.gpg.d/docker-com.gpg

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"

sudo apt update

apt-cache policy docker-ce

sudo apt install docker-ce

distribution=$(. /etc/os-release;echo $ID$VERSION_ID)

curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | sudo gpg --dearmor -o /etc/apt/trusted.gpg.d/nvidia-github-io-docker.gpg

curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.list | sudo tee /etc/apt/sources.list.d/nvidia-docker.list

curl -fsSL https://nvidia.github.io/libnvidia-container/gpgkey | sudo gpg --dearmor -o /etc/apt/trusted.gpg.d/nvidia-container-toolkit-keyring.gpg
curl -s -L https://nvidia.github.io/libnvidia-container/stable/deb/nvidia-container-toolkit.list | \
        sed 's#deb https://#deb [signed-by=/etc/apt/trusted.gpg.d/nvidia-container-toolkit-keyring.gpg] https://#g' | \
```

```
        sudo     tee     /etc/apt/sources.list.d/nvidia-container-
toolkit.list

    sudo apt-get update

    sudo apt-get install -y nvidia-container-toolkit nvidia-cuda-
toolkit

    sudo nvidia-ctk runtime configure --runtime=docker

    sudo nvidia-ctk runtime configure --runtime=containerd

    sudo systemctl restart docker

    sudo systemctl restart containerd

    sudo usermod -a -G docker terra_user # добавление пользователя
в группу docker, чтобы без sudo исполнять команды утилитой docker
```

Примечание. Если при установке ОС инсталлятор Ubuntu создал LVM раздел размером 200 Gb, вместо использования всего размера жесткого диска, необходимо выполнить следующие команды:

```
    sudo lvextend -l +100%FREE /dev/ubuntu-vg/ubuntu-lv
    sudo resize2fs /dev/ubuntu-vg/ubuntu-lv

    # проверка размеров разделов
    df -h
```

4) проверить доступность графического процессора из контейнера Docker через Nvidia-runtime выполнив следующую команду:

```
    sudo docker run --rm --runtime=nvidia --gpus all ubuntu
nvidia-smi
```

Ожидаемый вывод утилиты `nvidia-smi` представлен на рисунке 1:

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 535.154.05             Driver Version: 535.154.05   CUDA Version: 12.2     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  NVIDIA GeForce RTX 4090        Off | 00000000:01:00.0  On |                  Off |
|  0%   41C    P8              30W / 450W |      1MiB / 24564MiB |      0%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Рисунок 1. Ожидаемый вывод утилиты nvidia-smi

5) отключить автоматическое обновление пакетов в Ubuntu выполнив следующие команды:

```
sudo systemctl stop unattended-upgrades
sudo systemctl disable unattended-upgrades

sudo vi /etc/apt/apt.conf.d/20auto-upgrades
APT::Periodic::Update-Package-Lists "0"
APT::Periodic::Unattended-Upgrade "0"
```

5.2 Выпуск лицензии IVA Terra

Для выпуска лицензии IVA Terra необходимо выполнить перечисленную ниже последовательность действий:

1) подготовить отдельный стенд для выпуска лицензий выполнив следующие шаги:

– установить ОС Ubuntu версии 22.04 или выше;

– установить ПО Python версии 3.10 или выше, выполнив команды:

```
sudo apt update
sudo apt install python3
sudo apt install python3-pip
```

– установить пакет криптографии выполнив команду:

```
pip install PyCryptodome==3.20.0
```

– создать каталог выполнив команду:

```
mkdir TerraLicenseCenter
```

– в каталоге `TerraLicenseCenter` разместить файлы, находящиеся по ссылкам:

`https://git.hi-tech.org/terra/terra-core/-/blob/main/tests/examples/license_lifecycle/license_management.py`

`https://git.hi-tech.org/terra/terra-core/-/blob/main/rapi/terra_license.py`

– установить бит исполнения на файле управления лицензированием выполнив команду:

```
chmod a+x license_management.py
```

2) подготовить окружение для `LicenseManagement` выполнив следующие шаги:

– в каталоге `TerraLicenseCenter` создать каталог для размещения приватного ключа IVA Terra выполнив команду

```
$ mkdir terra-keys-dir
```

– в каталоге `TerraLicenseCenter/terra-keys-dir` разместить файл приватного ключа `TERRA_RSA_4096_2024_03_19_16_07_43.key.pem`;

– в каталоге `TerraLicenseCenter/terra-keys-dir` разместить файл публичного ключа `TERRA_RSA_4096_2024_03_19_16_07_43.key.pub.pem`.

3) сформировать клиентскую ключевую пару (публичного и приватного ключей) выполнив следующие шаги:

Примечание. Для клиентов, которые не предоставляют файл публичного ключа, формируется ключевая пара (связка ключей, состоящая из публичного и приватного ключей).

– запустить утилиту LicenseManagement с командой: create-client-key-pair, опционально можно указать каталог для сохранения ключей --client-keys-dir=[client-keys-dir], выполнив команду:

```
$ ./license_management.py create-client-key-pair --client-keys-dir=<client_name>-keys
```

где <client_name> – название компании (Клиента)

– в каталоге <client_name>-keys проверить наличие двух файлов содержащие приватный и публичный ключи:
TERRA_CLIENT_RSA_4096_2024_03_21_11_31_41.key.pem,
TERRA_CLIENT_RSA_4096_2024_03_21_11_31_41.key.pub.pem

Примечание. Путь к этому каталогу необходимо будет указывать при выполнении команды создания и подписи лицензии или команды проверки корректности подписи.

4) разместить файл публичного ключа, предоставленного клиентом, выполнив команды:

```
$ ls -la <client_name>-presented-key
-rw-rw-r-- 1 terra_user terra_user 799 Mar 21 11:31 from-<client_name>.key.pub.pem
```

Примечание. Если клиент предоставляет файл публичного ключа (файл используется для шифрования содержимого лицензии), то необходимо создать каталог выполнив команду $ mkdir <client_name>-presented-key и разместить в каталоге файл публичного ключа.

Примечание. Изменить имя файла публичного ключа так чтобы оно оканчивалось на суффикс «.key.pub.pem»

5) подготовить запрос на выпуск лицензии выполнив следующие шаги:

– запустить утилиту LicenseManagement с командой create-request, опционально можно указать каталог для сохранения файла выполнив команду:

```
$ ./license_management.py create-request --license-dir <client_name>)
```

– убедиться в наличии файла `license_project_2024_03_21_14_12_19.json` (в имени файла используется текущее системное время) в каталоге `<client_name>` (если каталог не был указан, будет создан каталог `license-dir`);

– открыть файл запроса на выпуск лицензии, и исправить значения в необходимых полях;

Пример содержимого файла запроса на выпуск лицензии:

```
{
"issuer_key": "Первый ключ выпуска Terra-лицензий ООО ХайТек",
"license_owner": "Отдел разработки. Использование в Terra-Core",
"begin_date": "2024-03-21",
"expire_date": "2024-09-17",
"worker_count": 2
}
```

6) выпуск лицензии проводится в следующей последовательности:

– убедиться, что в каталоге, предназначенном для выпуска лицензии содержится только один файл (`license_project_*********.json`);

– запустить утилиту `LicenseManagement` с командой `sign`, указав путь к каталогу в котором размещён файл запроса на выпуск лицензии `license_project_2024_03_21_14_12_19.json`, путь к каталогу, в котором содержится приватный ключ IVA Terra, и путь к каталогу в котором содержится публичный ключ Клиента:

```
$ ./license_management.py sign --license-dir <client_name> --terra-keys-dir terra-keys-dir --client-keys-dir <client_name>-keys/
Создан файл с открытым контентом <client_name>/TERRA_2024_03_21_12_18_37.license.open.json
Создан файл с шифрованным контентом <client_name>/TERRA_2024_03_21_12_18_37.license
Создан файл подписи <client_name>/TERRA_2024_03_21_12_18_37.license.signature
```

Примечание. По умолчанию, если команда `sign` запускается без опциональных ключей (`--license-dir`, `--terra-keys-dir`, `--client-keys-dir`), `LicenseManagement` использует следующие каталоги:

– `license-dir` – для поиска в нём файла запроса и сохранения файлов с результатами формирования лицензии (`.open.json`, `.license`, `.license.signature`);

– `terra-keys-dir` – для поиска в нём файл приватного ключа (`.key.pem`);

– `client-keys-dir` – для поиска в нём файла публичного ключа (`.key.pub.pem`).

7) проверить целостность лицензии выполнив следующие шаги:

Примечание. Процедура возможна только в случае, если для Клиента сформировали ключевую пару и у нас есть приватный ключ Клиента из этой пары, для расшифровки содержимого лицензии.

– запустить утилиту `LicenseManagement` с командой `check`, указав пути к каталогам содержащим файлы лицензии и подписи лицензии (`.license`, `.license.signature`), файл публичного ключа IVA Terra (`.key.pub.pem`), файл приватного ключа из пары ключей созданных для Клиента (`.key.pem`):

```
$ ./license_management.py check --license-dir <client_name> --terra-keys-dir terra-keys-dir --client-keys-dir <client_name>-keys/
Результат верификации подписи: Успешно
{
"issuer_key": "Первый ключ выпуска Terra-лицензий ООО ХайТек",
"license_owner": "<client_name>. Отдел продаж",
"begin_date": "2024-03-21",
"expire_date": "2024-12-31",
"worker_count": 20,
"number": "TERRA-2024-03-21-12-26-06"
}
```

Если приватный ключ Клиента отсутствует, будет выведено соответствующее сообщение:

```
$ ./license_management.py check --license-dir <client_name> --terra-keys-dir terra-keys-dir --client-keys-dir <client_name>-keys/
Traceback (most recent call last):
File "/home/terra_user/LaunchTerra/LicenseCenter/./license_management.py", line 103, in <module>
client_private_key_file = find_file(args.client_keys_dir, '.key.pem')
File "/home/terra_user/LaunchTerra/LicenseCenter/./license_management.py", line 34, in find_file
raise ValueError(f'NotFound {search_suffix}')
ValueError: NotFound .key.pem
```

8) артефакты Лицензии, которые передаются Клиенту:

– файл, содержащий шифрованное содержимое Лицензии (`<client_name>/TERRA_2024_03_21_12_26_06.license`) – для размещения на стороне Клиента в каталоге `TERRA_DATA/licenses`;

– файл, содержащий подпись по шифрованному содержимому Лицензии (`<client_name>/TERRA_2024_03_21_12_26_06.license.signature`) – для размещения на стороне Клиента в каталоге `TERRA_DATA/licenses`;

– файл приватного ключа, если для Клиента создавалась ключевая пара (`<client_name>-keys/TERRA_CLIENT_RSA_4096_2024_03_21_11_31_41.key.pem`) – для размещения на стороне Клиента в каталоге `TERRA_DATA/owner_private_keys`;

– файл публичного ключа Terra, для проверки подписи, созданной приватным ключом IVA Terra (`terra-keys-dir/TERRA_RSA_4096_2024_03_19_16_07_43.key.pub.pem`) – для размещения на стороне Клиента в каталоге `TERRA_DATA/terra_public_keys`.

5.3 Установка и настройка IVA Terra

Для установки и настройки IVA Terra необходимо выполнить перечисленную ниже последовательность действий:

1) подготовить архивы docker-образов IVA Terra

В состав архивов и инструментов входят следующие компоненты:

– `terra_offline_api.v0.12.0.0.img.gz` – обязательный архив;

– `terra_offline_worker.v0.12.0.0.img.gz` – обязательный архив;

– `terra_online_worker.v0.12.0.0.img.gz` – архив необходимый в случае, если планируется использование IVA Terra в онлайн-режиме показа субтитров;

– `terra_adp_client.v0.12.0.0.img.gz` – архив необходимый в случае, если планируется использование IVA Terra в режимах суммаризатора и протоколиста;

– `deploy.terra_v0.12.0.0.sh`, `docker-compose-v0.12.0.0.yml` – файлы необходимые для развертывания IVA Terra.

2) загрузить в Docker полученные образы выполнив команды:

```
docker load -i terra_offline_api.v0.12.0.0.img.gz
docker load -i terra_online_worker.v0.12.0.0.img.gz
docker load -i terra_offline_worker.v0.12.0.0.img.gz
docker load -i terra_adp_client.v0.12.0.0.img.gz
```

3) создать каталог, в котором будут храниться файлы IVA Terra (можно использовать корень домашнего каталога пользователя, от имени которого будут выполняться операции) и разместить в нём файлы развёртывания `deploy.terra_v0.12.0.0.sh, docker-compose-v0.12.0.0.yml`.

4) при необходимости внести изменения в содержимое файла `docker-compose-v0.12.0.0.yml` (например, для указания номера порта, отличающегося от 9001, который используется по умолчанию).

Примечание 1. Так как ПК IVA Terra не используется в режиме онлайн, то необходимо выполнить следующие действия:

– удалить из YML-файла секции, соответствующие сервисам `terra_online_worker` и `terra_online_db`;

– в сервисе `terra_api_service` для переменной `USE_ONLINE` указать значение `false`, и удалить строки - `terra_online_db, - terra_online_worker` расположенные под ключом `depends_on`.

Примечание 2. Если развёртываемый экземпляр IVA Terra не планируется для взаимодействия с ADP службами (протоколирование и суммаризация), то необходимо выполнить следующие действия:

– удалить из YML-файла секцию, соответствующую сервису `terra_adp_client_service`;

– в секции сервиса `terra_worker_service_1` для переменной `USE_ADP_SERVICES` установить значение `false`.

5) если на сервере уже исполняется другая версия IVA Terra, то необходимо выполнить следующие действия:

```
$ docker stop $(docker ps -aq)
$ docker rm $(docker ps -aq)
```

Примечание. Если предыдущая версия управляется через `docker-compose`, то остановку необходимо выполнить через интерфейс `docker-compose` выполнив команду:

```
$ docker-compose -f docker-compose-v0.X.0.0.yml down
```

6) для развёртывания релиза необходимо выполнить команду:

```
bash deploy.terra_v0.12.0.0.sh
```

7) для проверки наличия каталогов поддержки использования лицензии `(licenses, owner_private_keys, terra_public_keys)`, которые инференс-воркер (v0.12.0.0) создаёт при запуске в каталоге `TERRA_DATA`, необходимо выполнить следующие команды:

```
terra_user@terra-core:~/LaunchTerra/v0.12.0.0/TERRA_DATA$ ls -la
total 304
drwxrwxrwx 7 terra_user terra_user   4096 Aug 19 15:57 .
```

```
drwxrwxr-x 5 terra_user terra_user   4096 Aug 20 17:08 ..
drwxr-xr-x 2 terra_user terra_user   4096 Aug 28 12:37 licenses
drwxr-xr-x 2 terra_user terra_user  12288 Aug 21 12:29 minutes
drwxr-xr-x  2  terra_user  terra_user    4096  Aug  19  16:11
owner_private_keys
drwxr-xr-x 2 terra_user terra_user 278528 Aug 28 18:54 rawaudio
drwxr-xr-x  2  terra_user  terra_user    4096  Aug  19  16:11
terra_public_keys
```

Примечание. Инференс-воркер будет запущен в режиме «БЕЗ ЛИЦЕНЗИИ».

8) разместить 4 файла лицензий (подготовленные в соответствии с п. 5.2) в соответствующих каталогах, по образцу (вначале указаны каталоги на хосте запуска `LicenseManager`):

`<client_name>/TERRA_2024_03_21_12_18_37.license` в каталог – `./TERRA_DATA/license/TERRA_2024_03_21_12_18_37.license`

`<client_name>/TERRA_2024_03_21_12_18_37.license.signature` в каталог – `./TERRA_DATA/license/TERRA_2024_03_21_12_18_37.license.signature`

`<client_name>-keys/TERRA_CLIENT_RSA_4096_2024_03_21_11_31_41.key.pem` в каталог – `./TERRA_DATA/owner_private_keys/TERRA_CLIENT_RSA_4096_2024_03_21_11_31_41.key.pem`

`terra-keys-dir/TERRA_RSA_4096_2024_03_19_16_07_43.key.pub.pem` в каталог – `./TERRA_DATA/terra_public_keys/TERRA_RSA_4096_2024_03_19_16_07_43.key.pub.pem`

9) перезапустить IVA Terra выполнив команды:

```
docker-compose -f docker-compose-v0.12.0.0.yml down
docker-compose -f docker-compose-v0.12.0.0.yml up -d
```

10) проверить наличие информации о лицензии в логах IVA Terra (пример сообщений в Terra-Core - `https://<IP-адрес IVA Terra>/logs/7`):

```
2024-03-20 18:54:29,028 - root - INFO - Trying use
/var/www/alldata/licenses/TERRA_2024_03_19_16_10_36.license
VerifyWith:/var/www/alldata/terra_public_keys/TERRA_RSA_4096_
2024_03_19_16_07_43.key.pub.pem
```

```
DecryptWith:/var/www/alldata/owner_private_keys/TERRA_CLIENT_
RSA_4096_2024_03_19_16_10_03.key.pem
2024-03-20 18:54:29,092 - root - INFO - Найдена лицензия с
верифицированной подписью
Номер: TERRA-2024-03-19-16-10-36
Владелец: Отдел разработки. Использование в Terra-Core
Подпись: Первый ключ выпуска Terra-лицензий ООО ХайТек
Срок действия: от 2024-03-19 до 2024-09-15
```

11) Настроить сервер IVA MCU выполнив следующие шаги:

– войти в web-панель администрирования, перейти в раздел Системные настройки и выбрать секцию Настройки распознавания речи;

– в поле «Система распознавания речи» выбрать: `IVA Terra`;

– в поле «API URL системы распознавания речи» ввести: `https://<IP-terra>:<port>/process`;

– в поле «API Key системы распознавания речи» вести: `ivcs`;

– в поле «API URL формирования протокола собрания» ввести: `https://<IP-terra>:<port>/minutes`.

– выбрать секцию Настройки мероприятий и в поле «Запись стенограммы» включить флаговую кнопку;

– нажать кнопку Сохранить.

5.4 Проверка работы ПК IVA Terra

Для проверки работы ПК IVA Terra необходимо выполнить следующие шаги:

1) авторизоваться в платформе IVA MCU: войти в IVA WEB, в окне «Вход» ввести данные учетной записи IVA MCU и нажать кнопку «Войти»;

2) создать мероприятие: перейти в раздел «Мероприятия» и нажать кнопку «Начать сейчас»;

3) во время проведения мероприятия включить запись стенограммы: нажать кнопку  и выбрать «Запустить стенограмму»;

4) проговорить заранее подготовленный текст;

5) остановить запись стенограммы: нажать кнопку  и выбрать «Остановить стенограмму»;

6) перейти в разделе «Файлы»  и убедиться отображается файл со стенограммой который можно открыть и/л скачать.

6 ПРИЛОЖЕНИЕ

ОПИСАНИЕ ЗНАЧЕНИЙ ПЕРЕМЕННЫХ ОКРУЖЕНИЯ

1) сервис: `rabbitmq` – имя сервиса также используется как имя хоста в соседних сервисах внутри текущего YML-файла, для подключения к этому сервису

– `RABBITMQ_DEFAULT_USER: "guest"` – логин, который сервис принимает для подключения к себе;

– `RABBITMQ_DEFAULT_PASS: "guest"` – пароль, который сервис принимает для подключения к себе.

2) сервис: `terra_api_service`

– `OFFLINE_LANG_TO_TRANSLATE: ru` – код языка, будет использоваться для формирования пути к извлекаемому файлу протокола встречи, должен совпадать с такой же настройкой сервиса `terra_worker_service_1`;

– `TERRA_API_PORT: 9001` – номер порта, на котором сервис принимает подключения;

– `MINUTES_QUEUE_NAME: minutes_queue_9001` – имя очереди сообщений, в брокере сообщений, в которую будут отправляться сообщения воркеру;

– `LOGGING_LEVEL: INFO` – максимальный уровень отладочных сообщений, которые будут отправляться в файл лога (для отладки использовать значение `DEBUG`);

– `LOG_MAXFILESIZE: 10000000` – максимальный размер файла лога, после которого будет происходит создание нового файла и отправка в архив (переименование) предыдущего;

– `LOG_FILECOUNT: 10` – максимальное количество файлов архива;

– `LOG_WORKER_NAME: API_SRV` – суффикс имени файла лога

– `RABBITHost: rabbitmq` – имя хоста (или IP) брокера сообщений, в очереди которого будут отправляться сообщения;

– RABBITPort: 5672 – номер порта брокера сообщений;

– RABBITU: guest – логин для подключения к брокеру сообщений;

– RABBITPass: guest – пароль для подключения к брокеру сообщений;

– USE_ONLINE: true – выключатель поддержки онлайн-транскрибации, если выключен, то сервисы terra_online_db и terra_online_worker не будут задействованы в работе и режим субтитров будет недоступен клиенту;

– ONLINE_PROCESS_URL:"https://terra_online_worker:9977/process" – адрес онлайн воркера;

– ONLINE_DB_URL:"postgresql://terra_online:terra_online@terra_online_db:55432/terra_online_state" – адрес подключения к контейнеру хранения состояния;

– ONLINE_STORE_TYPE: "database" – тип хранилища, альтернативно доступен Redis, но в compose-файл нужно будет добавить сервис запуска контейнера Redis и поменять строку подключения ONLINE_DB_URL;

– CLEANING_PATTERNS: /var/www/alldata/cleaning_patterns.txt – путь к регулярным выражениям, удаляющим мусор, здесь используется единый путь с оффлайн воркером, но можно отнести в отдельный файл;

– MAX_DOUBLE_CHECKED_SOUND: 5.0 – максимальная длительность накопленного состояния (последовательность фрагментов одного спикера), при котором в онлайн воркер будет посылаться удвоенный звуковой контент;

– REMOVE_SHORT_SOUND: false – сброс состояния спикера (удаление аудиофрагментов), если в коротких фрагментах не был найден текст;

– TWICE_TEXT_IN_DOUBLE_SOUND: true – проверка наличия двойного текста в удвоенном звуковом контенте.

3) сервис: terra_worker_service_1

Примечание. terra_worker_service_1 уникальное имя сервиса, при размножении количества воркеров методом копирования необходимо исправить имя для достижения уникальности.

– NVIDIA_VISIBLE_DEVICES: 0 – через запятую указывается список индексов CUDA-устройств, которое будет доступно контейнеру;

– MINUTES_QUEUE_NAME: minutes_queue_9001 – имя очереди в брокере сообщений, из которой воркер забирается сообщения для обработки;

– CPUONLY: false – переключатель режима «использовать CPU вместо CUDA/GPU» (по умолчанию режим выключен, используется устройство CUDA, значение true включает режим использования CPU);

– CUDA_ID: 0 – индекс устройства (видеокарты), который виден библиотеке CUDA, из списка индексов, уже ограниченных настройкой NVIDIA_VISIBLE_DEVICES;

Примечание. Например, если NVIDIA_VISIBLE_DEVICES=0, 4; то CUDA_ID=0 – это устройство, которое видно в системе через nvidia-smi с индексом 0, а CUDA_ID=1 – это устройство, которое видно через nvidia-smi с индексом 4;

– PROMPT: "" – текстовая строка запроса инициализации, передаваемого в модель при загрузке;

Примечание. Непустая строка может быть причиной наличия двойного текста в удвоенном звуковом контенте передачи и потери текста;

– CPU_THREADS: 8 – количество потоков (задействованных ядер CPU), которое будет использовать PyTorch при работе с устройством CPU, оптимальное число 8;

– BATCH: 8 – размер пакета данных при отправке запросов в модель, оптимальное число 8 для обоих поддерживаемых типа устройств (CPU, CUDA);

– VAD_ON: 0.40 – минимальное значение голосовой активности, при которой звук будет распознан как речь;

– VAD_OFF: 0.25 – максимальное значение голосовой активности, при которой звук будет распознан как речь;

– COMBINE_SOUND_BY_PARTICIPANT: true – флаг группировки аудиофрагментов по спикеру перед отправкой звука на распознавание в модель, иначе весь звук будет объединён в единый звуковой файл и упорядочен по времени поступления фрагментов;

– USE_SILENCE_DELIMITER: true – вставка односекундных пауз тишины между соседними по времени аудиофрагментами разных спикеров, для избежания слияния текстовых фрагментов различных временных периодов в единые фразы;

– CLEANING_PATTERNS: /var/www/alldata/cleaning_patterns.txt – путь к файлу, содержащему регулярные выражения для фраз, исключаемых из итогового протокола.

Примечание. Файл размещается в каталоге, который смонтирован как раздел файловой системы в целевой системе DockerEngine, и доступен для редактирования.

Если файл изначально отсутствует, то он будет создан со списком значений по умолчанию, затем его содержимое можно отредактировать.

Если файл будет пустой, то весь текст возвращенный нейросетью будет содержаться в итоговом протоколе;

– OFFLINE_LANG_TO_TRANSLATE: ru – код языка, на который будет выполнен перевод и получен текст из модели (возможные коды: en, ru, es, ar, uz, tr);

– DETECT_LANG_IN_COMBINE: false – флаг перехода в режим обнаружения языка по содержимому комбинированного файла конференции, в таком случае итоговый текст будет на распознанном языке;

– COMPUTE_TYPE: FLOAT16 – тип данных, используемый при загрузке модели, актуально только при использовании устройства CUDA (возможные значения int8, float16, float32, float64);

– DAYS_STORE_AUDIO: 5 – количество дней хранения аудиофайлов, после которого они будут удалены;

– DAYS_STORE_MINUTES: 60 – количество дней хранения xml-протоколов конференций, после которого они будут удалены;

– LOGGING_LEVEL: INFO – максимальный уровень отладочных сообщений, которые будут отправляться в файл лога (для отладки использовать значение DEBUG);

– LOG_MAXFILESIZE: 10000000 – максимальный размер файла лога, после которого будет происходит создание нового файла, и отправка в архив (переименование) предыдущего;

– LOG_FILECOUNT: 10 – максимальное количество файлов архива;

– LOG_WORKER_NAME: 1 – суффикс имени файла лога, в который будут сохраняться сообщения этого воркера;

Примечание. Строку необходимо исправить для достижения уникальности.

– RABBITHost: rabbitmq – имя хоста (или IP) брокера сообщений, в очереди которого будут отправляться сообщения;

– RABBITPort: 5672 – номер порта брокера сообщений;

– RABBITU: guest – логин для подключения к брокеру сообщений;

– RABBITPass: guest – пароль для подключения к брокеру сообщений;

– USE_ADP_SERVICES: true – переключатель режима общения с terra_adp_client_service;

– ADP_CLIENT_URL:http://terra_adp_client_service:9019/extend_minutes – адрес эндпоинта обработки запроса на обогащение протокола встречи списком поручений и суммаризацией после общения с ADP сервисами;

– FORCE_EXTEND_MINUTES: true – отправка дополнительного флага на адрес ADP_CLIENT_URL о необходимости выполнения обработки протокола встречи, даже если он уже имеет статус обработанного.

4) сервис: terra_online_worker

Примечание. terra_online_worker уникальное имя сервиса, при размножении количества воркеров методом копирования необходимо исправить имя для достижения уникальности.

– NVIDIA_VISIBLE_DEVICES: 0 – через запятую указывается список индексов CUDA-устройств, которое будет доступно контейнеру;

– CPUONLY: false – переключатель режима «использовать CPU вместо CUDA/GPU»

Примечание. По умолчанию режим выключен, используется устройство CUDA, значение true включает режим использования CPU.

– CUDA_ID: 0 – индекс устройства (видеокарты), который виден библиотеке CUDA, из списка индексов, уже ограниченных настройкой NVIDIA_VISIBLE_DEVICES.

Примечание. Например, если NVIDIA_VISIBLE_DEVICES=0, 4; то CUDA_ID=0 - это устройство которое видно в системе через nvidia-smi с индексом 0, а CUDA_ID=1 это устройство которое видно через nvidia-smi с индексом 4;

– PROMPT: "" – текстовая строка запроса инициализации, передаваемого в модель при загрузке;

Примечение. Непустая строка может быть причиной наличия двойного текста в удвоенном звуковом контенте передачи и потери текста;

– CPU_THREADS: 8 – количество потоков (задействованных ядер CPU), которое будет использовать PyTorch при работе с устройством CPU, оптимальное число «8»;

– BATCH: 8 – размер пакета данных при отправке запросов в модель, оптимальное число 8, для обоих поддерживаемых типа устройств (CPU, CUDA);

– VAD_ON: 0.40 – минимальное значение голосовой активности, при которой звук будет распознан как речь;

– VAD_OFF: 0.25 – максимальное значение голосовой активности, при которой звук будет распознан как речь;

– OFFLINE_LANG_TO_TRANSLATE: ru – код языка, на который будет выполнен перевод и получен текст из модели (возможные коды: en, ru, es, ar, uz, tr);

– COMPUTE_TYPE: FLOAT16 – тип данных, используемый при загрузке модели, актуально только при использовании устройства CUDA (возможные значения int8, float16, float32, float64);

– LOGGING_LEVEL: INFO – максимальный уровень отладочных сообщений, которые будут отправляться в файл лога (для отладки использовать значение DEBUG);

– LOG_MAXFILESIZE: 10000000 – максимальный размер файла лога, после которого будет происходит создание нового файла, и отправка в архив (переименование) предыдущего;

– LOG_FILECOUNT: 10 – максимальное количество файлов архива;

– LOG_WORKER_NAME: OL_WORKER_1 – суффикс имени файла лога, в который будут сохраняться сообщения этого воркера;

Примечание. Строку необходимо исправить для достижения уникальности.

5) Сервис terra_online_db

– POSTGRES_USER: "terra_online" – логин, который сервис принимает для подключения к себе;

– POSTGRES_PASSWORD: "terra_online" – пароль, который сервис принимает для подключения к себе;

– POSTGRES_DB: "terra_online_state" – имя базы данных в СУБД.

6) сервис: terra_adp_client_service

Примечание. terra_adp_client_service уникальное имя сервиса, при размножении количества воркеров методом копирования необходимо исправить имя для достижения уникальности.

– LOGGING_LEVEL: INFO – максимальный уровень отладочных сообщений которые будут отправляться в файл лога (для отладки использовать значение DEBUG);

– LOG_MAXFILESIZE: 10000000 – максимальный размер файла лога после которого будет происходит создание нового файла, и отправка в архив (переименование) предыдущего;

– LOG_FILECOUNT: 10 – максимальное количество файлов архива;

– LOG_WORKER_NAME: ADP_CLIENT – суффикс имени файла лога, в который будут сохраняться сообщения этого воркера;

Примечание. Строку необходимо исправить для достижения уникальности.

– CLIENT_PORT: 9019 – номер порта на котором сервис принимает подключения;

– ADP_QUESTION_URL: http://terra.iva.ru:9001/squestion – адрес эндпоинта принимающего запросы на обработку;

– ADP_QUESTION_TOKEN: ivcs – токен авторизации на эндпоинте ADP_QUESTION_URL;

– SECONDS_IN_QUESTION: 600 – максимальная длительность текстовой посылки, в секундах

– QUESTION_TIMEOUT: 2.0 – максимальная длительность ожидания ответа от ADP_QUESTION_URL, в секундах;

– ADP_ANSWER_URL: http://terra.iva.ru:9001/sanswer/ – адрес эндпоинта принимающего запросы на получение результатов обработки;

– ANSWER_TIMEOUT: 20 – максимальная длительность ожидания ответа от ADP_ANSWER_URL, в секундах;

– ANSWER_ATTEMPT: 15 – максимальное количество попыток получения результатов обработки от эндпоинта ADP_ANSWER_URL;

– PLANNER_PARTICIPANT_ID: '00000000' – UUID идентификатор спикера, который будет использован при добавлении в протокол встречи фразы содержащей список поручений, возвращённый от ADP сервиса planner;

– SUMMARY_PARTICIPANT_ID: '11111111' – UUID идентификатор спикера, который будет использован при добавлении в протокол встречи фразы содержащей краткое изложение встречи, возвращённый от ADP сервиса summary.