УТВЕРЖДЕН
41327606.425000.462-01 33 01-ЛУ

Руководство программиста
«Speech - обработчик речевой информации»
(подсистема проекта SOVA)

41327606.425000.462-01 33 01
Листов 21

2020

Инв. № подл.	Подпись и дата	Взам. инв. №	Инв. № дубл.	Подпись и дата

АННОТАЦИЯ

В данном программном документе приведено руководство программиста по настройке

И использованию специального программного обеспечения «Speech».

В данном программном документе, в разделе «Назначение и условия применения программы» указаны назначение и функции, выполняемые программой, условия, необходимые для выполнения программы (объем оперативной памяти, требования к составу и параметрам периферийных устройств, требования к программному обеспечению и т.п.).

В разделе «Характеристика программы» приведено описание основных характеристик и особенностей программы (режим работы, средства контроля правильности выполнения и самовосстанавливаемости программы и т.п.).

В данном программном документе, в разделе «Входные и выходные данные» приведено описание организации используемой входной и выходной информации.

В разделе «Сообщения» указаны тексты сообщений, выдаваемых программисту или оператору в ходе выполнения программы, описание их содержания и действий, которые необходимо предпринять по этим сообщениям.

Оформление программного документа «Руководство программиста» произведено по требованиям ЕСПД (ГОСТ 19.101-77 1), ГОСТ 19.103-77 2), ГОСТ 19.104-78* 3), ГОСТ 19.105-78* 4), ГОСТ 19.106-78* 5), ГОСТ 19.504-79* 6), ГОСТ 19.604-78* 7)).

СОДЕРЖАНИЕ

1 НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ ПРОГРАММЫ

1.1 Назначение программы

Программа должна использоваться в системах автоматизации диалога - ботах для предоставления необходимой информации клиентам контактных центров.

Конечными пользователями программы могут являться как сотрудники предприятий или контактных центров (полное право доступа к информации), так и лица, обратившиеся на предприятие по каналам связи.

Возможна интеграция продукта в сторонние системы при помощи интерфейса API или применение в качестве его локального или сетевого модуля для использования в сетевой инфраструктуре.

1.2 Функции, выполняемые программой

Программный модуль распознавания речи SOVA ASR предназначен для перевода аудиозаписей, содержащих речь на русском языке, в текст, обеспечивая высокий уровень производительности и качества распознавания речи.

Программный модуль синтеза речи SOVA TTS предназначен для генерации аудио, содержащего озвученный голосом выбранного диктора переданный текст на русском языке, при этом модуль обеспечивает высокий уровень производительности и качества синтезируемой речи.

1.3 Условия, необходимые для выполнения программы

Решение должно быть развёрнуто на ЭВМ с техническими характеристиками не ниже минимальных требований и операционной системой, поддерживающей технологии Docker, docker-compose, CUDA, cuDNN.

1.3.1 Минимальные требования:

- тактовая частота процессора – от 2 ГГц,
- архитектура x86_64,
- количество ядер процессора – от 8 штук,
- от 16 Гб оперативной памяти,
- размер видео памяти – от 8 Гб,
- поддержка CUDA,
- размер дисковый накопитель – от 50 Гб

1.3.2 Рекомендуемые параметры оборудования

Для устойчивой работы программы рекомендуется использовать ЭВМ с характеристиками не хуже:

- Intel(R) Xeon(R) Platinum 8259CL CPU @ 2.50GHz, x86_64, 8 cores
- 32 GB RAM
- NVIDIA Tesla T4

Аудиофайлы, подаваемые на вход системе распознавания речи SOVA ASR, должны удовлетворять требованиям к формату, частоте дискретизации и длительности аудио. При отличии данных параметров корректная работа модуля возможна, но не гарантирована.

Система синтеза речи SOVA TTS принимает на вход текст на русском языке, содержащий любые кириллические буквы, а также определённый набор знаков препинания и специальных символов:

- «.» - точка;
- «,» - запятая;
- «?» - вопросительный знак;
- « - » - тире, обрамлённое пробелами;
- «+» - плюс (для пользовательских ударений).

При вводе любых других символов и знаков препинания программа будет игнорировать их, при вводе же букв, отличных от букв кириллического алфавита, качественная работа программы не гарантируется.

1.4 Требования к составу и параметрам периферийных устройств

Необходимы устройства ввода (клавиатура, мышь) и вывода (монитор) для взаимодействия с модулями SOVA ASR и SOVA TTS. Для разворачивания дистрибутива необходима возможность подключения устройства, на котором дистрибутив передаётся в случае передачи на носителе (USB порт для флеш накопителя или дисковод). Дополнительных требований не предъявляется.

1.5 Требования к программному обеспечению

Для разворачивания модулей SOVA ASR и SOVA TTS необходимо предварительное выполнение требований к ПО:

Операционная система, поддерживающая технологию Docker, docker-compose, CUDA, cuDNN, архитектуру x86_64.

Работоспособность программы может быть гарантирована на системе, укомплектованной нижеуказанными версиями ПО или более новыми:

- Ubuntu version: .....18.04.3 LTS x86_64 GNU/Linux
- Docker version: .....19.03.8, build afacb8b7f0
- docker-compose version: 1.25.5, build 8a1c60f6
- nvcc version: .....release 10.1, V10.1.243
- Driver version: .....440.100 CUDA Version: 10.1

1.6 Порядок развертывания модуля SOVA ASR

Для развертывания модуля SOVA ASR необходимо выполнить следующие шаги[1]:

1.6.1 Установите Docker и docker-compose:

```
 $ sudo apt-get update
 $ sudo apt-get install \
      apt-transport-https \
      ca-certificates \
      curl \
      gnupg-agent \
      software-properties-common
 $ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
 $ sudo apt-key fingerprint 0EBFCD88
 $ sudo add-apt-repository \
    "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
    $(lsb_release -cs) \
    stable"
 $ sudo apt-get update
 $ curl -s -L https://nvidia.github.io/nvidia-container-runtime/gpgkey | \
   sudo apt-key add -
 $ distribution=$(. /etc/os-release;echo $ID$VERSION_ID)
 $ curl -s -L https://nvidia.github.io/nvidia-container-
runtime/$distribution/nvidia-container-runtime.list | \
   sudo tee /etc/apt/sources.list.d/nvidia-container-runtime.list
 $ sudo apt-get update
 $ sudo apt-get install docker-ce docker-ce-cli containerd.io nvidia-container-
runtime
 $ sudo curl -L "https://github.com/docker/compose/releases/download/1.25.5/docker-
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
 $ sudo chmod +x /usr/local/bin/docker-compose
```

[1] Инструкция приводится для развертывания модуля в среде Ubuntu 18.04

1.6.2 Отредактируйте файл /etc/docker/daemon.json

Отредактируйте/создайте файл **/etc/docker/daemon.json** в удобном текстовом редакторе с правами суперпользователя:

```
{
        "runtimes": {
                "nvidia": {
                        "path": "nvidia-container-runtime",
                        "runtimeArgs": []
                }
        },
        "default-runtime": "nvidia"
}
```

1.6.3 Перезапустите сервис:

```
$ sudo systemctl restart docker.service
```

1.6.4 Разверните репозиторий

Разверните репозиторий (по умолчанию путь к репозиторию - **/opt/sova-asr**, при отсутствии доступа к интернету возможна установка с внешнего накопителя путём копирования файлов репозитория), скопируйте веса моделей в внутреннюю папку data (либо другую удобную, путь указывается в конфигурационных файлах):

```
$ cd /opt/
$ sudo git clone --recursive https://github.com/sovaai/sova-asr.git
 sova-asr
```

1.6.5 Соберите образ

```
$ cd /opt/sova-asr/
$ sudo docker-compose build
```

1.6.6 Запустите контейнер с приложением:

```
$ sudo docker-compose up -d
```

После внесения изменений в коде (не Dockerfile), обновления весов моделей либо для восстановления работоспособности сервиса – перезапустите докер контейнер командой для перезапуска приложения:

```
$ sudo docker-compose up -d --force-recreate
```

1.7 Порядок развертывания модуля SOVA TTS

Для развертывания модуля SOVA TTS необходимо выполнить следующие шаги[2]:

[2] Инструкция приводится для развертывания модуля в среде Ubuntu 18.04

1.7.1 Установите Docker и docker-compose:

(Шаг аналогичен 1.6.1 и при установке на одну и ту же ЭВМ повторного исполнения не требует)

```
 $ sudo apt-get update
 $ sudo apt-get install \
      apt-transport-https \
      ca-certificates \
      curl \
      gnupg-agent \
      software-properties-common
 $ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
 $ sudo apt-key fingerprint 0EBFCD88
 $ sudo add-apt-repository \
    "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
    $(lsb_release -cs) \
    stable"
 $ sudo apt-get update
 $ curl -s -L https://nvidia.github.io/nvidia-container-runtime/gpgkey | \
   sudo apt-key add -
 $ distribution=$(. /etc/os-release;echo $ID$VERSION_ID)
 $ curl -s -L https://nvidia.github.io/nvidia-container-
runtime/$distribution/nvidia-container-runtime.list | \
   sudo tee /etc/apt/sources.list.d/nvidia-container-runtime.list
 $ sudo apt-get update
 $ sudo apt-get install docker-ce docker-ce-cli containerd.io nvidia-container-
runtime
 $ sudo curl -L "https://github.com/docker/compose/releases/download/1.25.5/docker-
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
 $ sudo chmod +x /usr/local/bin/docker-compose
```

1.7.2 Отредактируйте файл /etc/docker/daemon.json

(Шаг аналогичен 1.6.2 и при установке на одну и ту же ЭВМ повторного исполнения не требует)

Отредактируйте/создайте файл **/etc/docker/daemon.**json в удобном текстовом редакторе с правами суперпользователя:

```
{
        "runtimes": {
                "nvidia": {
                        "path": "nvidia-container-runtime",
                        "runtimeArgs": []
                }
        },
        "default-runtime": "nvidia"
}
```

1.7.3 Перезапустите сервис:

(Шаг аналогичен 1.6.3 и при установке на одну и ту же ЭВМ повторного исполнения не требует)

```
 $ sudo systemctl restart docker.service
```

1.7.4 Разверните репозиторий

Разверните репозиторий (по умолчанию путь к репозиторию - **/opt/sova-tts**, при отсутствии доступа к интернету возможна установка с внешнего накопителя путём копирования файлов

репозитория), скопируйте веса моделей в папку data (либо другую удобную, путь указывается в конфигурационных файлах)::

```
$ cd /opt/
$ sudo git clone --recursive https://github.com/sovaai/sova-tts.git
 sova-tts
```

1.7.5 Соберите образ

```
$ cd /opt/sova-tts/
$ sudo docker-compose build
```

1.7.6 Запустите контейнер с приложением:

```
$ sudo docker-compose up -d
```

После внесения изменений в коде (не Dockerfile), обновления весов моделей либо для восстановления работоспособности сервиса – перезапустите докер контейнер командой для перезапуска приложения:

```
$ sudo docker-compose up -d --force-recreate
```

1.8 Требования к персоналу (программисту)

Минимальное количество персонала, требуемого для работы программы, должно составлять не менее 2 штатных единиц - системный программист и конечный пользователь программы - оператор.

Системный программист должен иметь техническое образование. В перечень задач, выполняемых системным программистом, должны входить:

1) задача поддержания работоспособности технических средств;

2) задачи установки (инсталляции) и поддержания работоспособности системных программных средств - операционной системы;

3) задача установки (инсталляции) программы.

2 ХАРАКТЕРИСТИКА ПРОГРАММЫ

2.1 Описание основных характеристик программы

Модуль SOVA ASR обеспечивает перевод аудио в текст с ошибками WER и CER не превышающими в среднем значения 30 и 15 соответственно.

Модуль SOVA TTS обладает следующими характеристиками:

1. Синтезированная речь не имеет искажений, шумов и прочих артефактов, мешающих нормальному восприятию сгенерированной речи;

2. Система позволяет регулировать высоту тона (увеличение на 50% и уменьшение на 25%) и скорость воспроизведения (ускорение/замедление до двух раз) сгенерированной речи;
3. Система обладает контролем ударений на базе словарей, а также предоставляет пользователю возможность управлять ударениями;
4. Система имеет на выбор два голоса: мужской и женский.

2.2 Временные характеристики программы

Модуль SOVA ASR обеспечивает перевод аудио в текст за время, в среднем не превышающее длину аудио при обеспечении требуемых технических характеристик ЭВМ.

Модуль SOVA TTS проводит генерацию речи за время в среднем не превышающее 3 секунд на текст из 100 символов при соблюдении выполнения минимальных технических характеристик сервера, на котором развёрнуто решение

2.3 Режим работы программы

Модули SOVA ASR и SOVA TTS работают в одном режиме, запуск с изменёнными настройками, в том числе альтернативными моделями возможен после изменения файлов моделей или конфигурационных файлов.

2.4 Средства контроля правильности выполнения программы

Программные модули являются REST API приложениями, правильность работы которых возможно проверить отправив запросы с тестовыми аудио файлами или текстовыми строками.

Пример запроса к SOVA ASR с локальной машины:

```
$ curl --location --request POST 'http://localhost:8888/asr/' \
    --form 'audio_blob=@"/home/ubuntu/test.wav"'
```

Пример ответа:

```
{
    "r": [{
      "response_audio_url": "/media/0b9001cd-d4ac-4402-90f3-75cb31e5b1d0.wav",
      "response_code": 0,
      "response": [{
            "text": "привет",
            "time": 0.478
      }]
```

```
        }]
    }
```

Пример запроса к SOVA TTS с локальной машины:

```
    $ curl --location --request POST 'http://localhost:8899/tts/' --form
'voice=Ruslan' --form 'text=привет'
```

Пример ответа:

```
    {
        "response": [
          {
                "name": "Ruslan",
                "time": 0.239
          }
        ],
        "response_audio": "…",
        "response_audio_url":     "/media/data/waves/ruslan_38bfb2f9d9a5ca_2020-08-
13_16-50.wav",
        "response_code": 0
    }
```

2.5 Описание основных особенностей программы

2.5.1 Самовосстанавливаемость программы

Программные модули восстанавливаются после нарушений функционирования стандартными средствами операционной системы и Docker.

3 ОБРАЩЕНИЕ К ПРОГРАММЕ

3.1 Описание процедур вызова программы

Для запуска сервиса модуля SOVA ASR, либо SOVA TTS перейдите в директорию с развёрнутым модулем и выполните команду:

```
 $ sudo docker-compose up -d
```

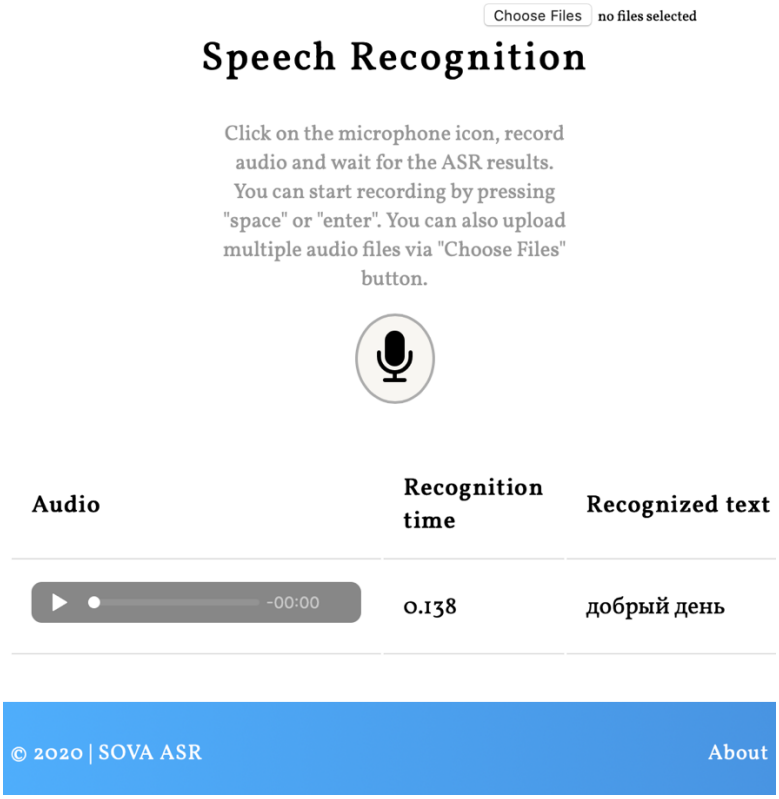
После внесения изменений в коде (не Dockerfile), обновления весов моделей либо для восстановления работоспособности сервиса – перезапустите докер контейнер командой для перезапуска приложения:

```
$ sudo docker-compose up -d --force-recreate
```

Запущенный сервис принимает POST-запросы на порты 8888 для SOVA ASR, 8899 для SOVA TTS, примеры запросов описаны в пункте 2.4.

3.2 Способы передачи управления и параметров данных программы

После запуска сервиса можно в браузере зайти в веб интерфейс по адресу http://localhost:8888 для SOVA ASR и http://localhost:8899 для SOVA TTS.



4 ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

4.1 Описание организации входной и выходной информации

4.1.1 Организация входных и выходных данных модуля SOVA ASR:

Входные данные модуля распознавания речи SOVA ASR представляют собой аудиофайлы формата wav со следующими характеристиками:

- Format .....Wave
- Overall bit rate mode.....Constant

- Overall bit rate.....256 kbps
- Channels.....1 channel
- Sampling rate .....16 kHz
- Bit depth.....16 bits

Длительность фрагментов аудио не должна быть меньше 100 мс и не должна превышать 1 минуту. Корректная работа модуля SOVA ASR с аудиофайлами большей длины возможна, но не гарантируется.

При необходимости распознавания речи в аудиофайлах иного формата (например, mp3/ogg/webm/m4a) или при другой частоте дискретизации (например, 8 кГц) есть возможность отправить их на вход системы, при этом будет предпринята попытка системы автоматически конвертировать их в нужный формат средствами ffmpeg.

Выходная информация представляет собой текстовую строку, содержащую символы русского алфавита, дефисы, пробелы и цифры.

Пример запроса к API модуля SOVA ASR:

```
"request": {
    "auth": {
        "type": "noauth"
    },
    "method": "POST",
    "body": {
        "mode": "formdata",
        "formdata": [
                {
                        "key": "audio_blob",
                        "type": "file",
                        "src": ""
                }
        ]
    },
    "url": {
        "raw": "http://localhost:8888/asr/",
        "protocol": "http",
```

```
        "host": [
                "localhost"
        ],
        "port": "8888",
        "path": [
                "asr",
                ""
        ]
    }
}
```

Формат ответа:

```
"response": {
    "r": [{
        "response_audio_url": "/media/0b9001cd-d4ac-4402-90f3-75cb31e5b1d0.wav",
        "response_code": 0,
        "response": [{
                "text": "мама мыла раму",
                "time": 0.478
        }]
    }]
}
```

4.1.2 Организация входных и выходных данных модуля SOVA TTS:

Входные данные модуля синтеза речи SOVA TTS представляют собой текст, содержащий символы из допустимого набора, перечисленного в пункте 4.3 Технического Задания. Размер беспробельного текста не должен превышать длину в 100 символов, иначе произойдёт сбой работы системы.

Выходные данные представляют собой аудиофайлы формата wav со следующими характеристиками:

- Format .....Wave
- Overall bit rate mode.....Constant
- Overall bit rate.....354 kbps

- Channels.....1 channel
- Sampling rate .....22.05 kHz
- Bit depth.....16 bits

Пример запроса к API модуля SOVA TTS:

```
"request": {
    "method": "POST",
    "body": {
        "mode": "formdata",
        "formdata": [
                {
                        "key": "voice",
                        "value": "Ruslan",
                        "type": "text"
                },
                {
                        "key": "text",
                        "value": "Привет, мир.",
                        "type": "text"
                }
        ]
    },
    "url": {
        "raw": "http://localhost:8899/tts/",
        "protocol": "http",
        "host": [
                "localhost"
        ],
        "port": "8899",
        "path": [
                "tts",
                ""
        ]
    }
```

```
}
```

Формат ответа:

```
{
    "response":[
        {
            "name": "Ruslan",
            "time": 0.778
        }
    ],
    "response_audio": …,
    "response_audio_url":"/media/data/waves/ruslan_38be3647b0931e_2020-08-11_19-25.wav",
    "response_code": 0
}
```

Пример запроса на Python с сохранение получаемого аудио блоба как файла:

```
import json
import requests
from base64 import b64decode


def tts(text):
    url = "http://localhost:8899/tts/"
    payload = {"voice": "Ruslan", "text": text}

    response = requests.request("POST", url, headers={}, data=payload)

    if response:
        return b64decode(json.loads(response.text)["response_audio"].encode("utf-8"))

    else:
        return False
```

```
def main():
    text = "д+обрый д+ень"
    audio = tts(text)
    with open("result.wav", "wb") as f:
        f.write(audio)


if __name__=="__main__":
    main()
```

4.2 Описание кодирования информации

5 СООБЩЕНИЯ

Программные модули запускаются внутри докер контейнеров и подробное логгирование остаётся внутри контейнеров, просмотреть логи можно узнав ID контейнера использовав команду **docker ps**, после чего выполнить команду **docker logs** *id_контейнера*. В выводе предусмотрены сообщения для отладки приложения в случае возникновения ошибок, которые может разрешить системный программист.

6 ПЕРЕЧЕНЬ ТЕРМИНОВ И СОКРАЩЕНИЙ

- ASR (Automatic Speech Recognition) — автоматическое распознавание речи;
- Speech язык разметки для приложений синтеза речи;
- TTS (Text To Speech) - технология, которая позволяет преобразовывать текстовые файлы в аудиофайлы при помощи анализа грамматических структур текста;
- KenLM - это быстрый набор инструментов для моделирования языка с низким объемом памяти, который масштабируется до триллионов слов;
- CPU - Центральный процессор — электронный блок либо интегральная схема, исполняющая машинные инструкции (код программ), главная часть аппаратного обеспечения компьютера или программируемого логического контроллера;
- Нейронная сеть также искусственная нейронная сеть, ИНС — математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей - сетей нервных клеток живого организма;

- Лексико-смысловой анализатор - система автоматического понимания текстов, заключающаяся в выделении семантических отношений, формировании семантического представления текстов;

- API - (программный интерфейс приложения, интерфейс прикладного программирования) (англ. application programming interface) — описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой,

- CUID- уникальный идентификатор чата, используется для идентификации чата, к которому относится очередной запрос СУБД

- Руководство системного программиста - перечень данных включающий в себя: общие сведения о программе, структуру программы, настройку программы, проверку программы, дополнительные возможности и сообщения системному программисту выдаваемых в ходе выполнения настройки, проверки программы, а также в ходе выполнения программы, описание их содержания и действий, которые необходимо предпринять по этим сообщениям;

- Руководство оператора - перечень данных, включающий в себя: назначение программы, условия выполнения программы, выполнение программы, сообщения оператору;

- Программа и методики испытаний - организационно-методический документ, обязательный к выполнению, включающий метод испытаний, средства и условия испытаний, алгоритмы выполнения операций по определению одной или нескольких взаимосвязанных характеристик свойств объекта, формы представления данных и оценивания точности, достоверности результатов;

- Ведомость эксплуатационных документов - учетный документ, составленный в виде списка эксплуатационных документов;

- ГОСТ- Межгосударственный стандарт — региональный стандарт, принятый Межгосударственным советом по стандартизации, метрологии и сертификации Содружества независимых государств;

- Техническое задание - перечень требований, условий, целей, задач, поставленных заказчиком в письменном виде, документально оформленных и выданных исполнителю работ проектно-исследовательского характер.

Оформление настоящего документа произведено в соответствии с ГОСТ 19.505 79.

7 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Shen, Jonathan, Ruoming Pang, Ron J. Weiss, Michael Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, R. J. Skerry-Ryan, Rif A. Saurous, Yannis Agiomyrgiannakis and Yonghui Wu. “Natural TTS Synthesis by Conditioning Wavenet on MEL Spectrogram Predictions.” *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2018): 4779-4783.
2. Prenger, Ryan, Rafael Valle and Bryan Catanzaro. “Waveglow: A Flow-based Generative Network for Speech Synthesis.” *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2019): 3617-3621.
3. Wang, Yuxuan, Daisy Stanton, Yu Zhang, R. J. Skerry-Ryan, Eric Battenberg, Joel Shor, Ying Xiao, Fei Ren, Ye Jia and Rif A. Saurous. “Style Tokens: Unsupervised Style Modeling, Control and Transfer in End-to-End Speech Synthesis.” *ArXiv* abs/1803.09017 (2018).
4. Zhu, Xiaolian, Y. Zhang, Shan Yang, Liumeng Xue and L. Xie. “Pre-Alignment Guided Attention for Improving Training Efficiency and Model Stability in End-to-End Speech Synthesis.” *IEEE Access* 7 (2019): 65955-65964.
5. Liu, Peng, Xixin Wu, Shiyin Kang, G. Li, Dan Su and Dong Yu. “Maximizing Mutual Information for Tacotron.” *ArXiv* abs/1909.01145 (2019).
6. Vineel Pratap, Awni Hannun, Qiantong Xu, Jeff Cai, Jacob Kahn, Gabriel Synnaeve, Vitaliy Liptchinsky, Ronan Collobert, “wav2letter++: The Fastest Open-source Speech Recognition System”, arXiv:1812.07625, 2018.
7. Ronan Collobert, Christian Puhrsch, and Gabriel Synnaeve, “Wav2letter: an end-to-end convnet-based speech recognition system,” *CoRR*, vol. abs/1609.03193, 2016.
8. Vitaliy Liptchinsky, Gabriel Synnaeve, and Ronan Collobert, “Letter-based speech recognition with gated convnets,” *CoRR*, vol. abs/1712.09444, 2017.
9. Matteo Frigo and Steven G. Johnson, “The design and implementation of FFTW3,” Proceedings of the IEEE, vol. 93, no. 2, pp. 216–231, 2005, Special issue on “Program Generation, Optimization, and Platform Adaptation”.
10. Alex Graves, Santiago Ferna´ndez, Faustino Gomez, and Ju¨rgen Schmidhuber, “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks,” in Proceedings of the 23rd international conference on Machine learning. ACM, 2006, pp. 369–376.

Лист регистрации изменений									
Номера листов (страниц)					Всего листов (страниц) в докум.	№ документа	Входящий № сопроводит. докум. и дата	Подп.	Дата
Изм.	измененных	замененных	новых	аннулированных					