

tts.py

tts.py

```
import warnings
warnings.filterwarnings("ignore", category=UserWarning)

import os
import sys
import torch
from http.server import BaseHTTPRequestHandler, HTTPServer
import json
import re
from num2words import num2words

language = 'ru'
device = torch.device('cpu')
model_path = "./v5_ru.pt"

print("Инициализация стабильной локальной модели Silero...")
if not os.path.exists(model_path):
    print("КРИТИЧЕСКАЯ ОШИБКА: Файл v5_ru.pt не найден!")
    sys.exit(1)

model =
torch.package.PackageImporter(model_path).load_pickle("tts_models",
"model")
model.to(device)
print("ИИ-процессор звука полностью готов к работе!")

# Расширенная база для чтения латиницы
# Умный словарь частых слов (чтобы читались красиво, а не просто побуквенно)
ENG_PHONETIC = {
    'hello': 'хэллоу', 'hi': 'хай', 'ai': 'эй', 'js': 'джей эс', 'php':
'пи эйч пи',
    'python': 'пайтон', 'google': 'гугл', 'ok': 'окей', 'yes': 'йес',
'no': 'ноу',
    'javascript': 'джаваскрипт', 'node.js': 'ноуд джей эс', 'node':
'ноуд',
    'apple': 'эпл', 'microsoft': 'майкрософт', 'windows': 'виндоус',
'server': 'сервер',
    'code': 'код', 'project': 'проджект', 'file': 'файл', 'text':
'текст'
}

# Карта перевода отдельных английских букв и буквосочетаний в русские слоги
ENGLISH_TO_RUSSIAN_ABC = {
    'ch': 'ч', 'sh': 'ш', 'th': 'ф', 'ph': 'ф', 'ee': 'и', 'oo': 'у',
    'ea': 'и',
```

```

    'a': 'э', 'b': 'б', 'c': 'к', 'd': 'д', 'e': 'е', 'f': 'ф', 'g': 'г',
    'h': 'х',
    'i': 'ай', 'j': 'дж', 'k': 'к', 'l': 'л', 'm': 'м', 'n': 'н', 'o':
    'о', 'p': 'п',
    'q': 'кь', 'r': 'р', 's': 'с', 't': 'т', 'u': 'ю', 'v': 'в', 'w': 'в',
    'x': 'икс',
    'y': 'ай', 'z': 'з'
}

def load_config():
    default_config = {"voice_settings": {"speaker": "baya", "speed":
1.0}, "word_replacements": {}}
    if os.path.exists('config.json'):
        try:
            with open('config.json', 'r', encoding='utf-8') as f:
                return json.load(f)
        except: return default_config
    return default_config

def clean_and_normalize_text(text, config):
    text_clean = text.strip()

    # 1. Применяем замену пунктуации
    for old_punc, new_punc in config.get("custom_punctuation",
{}).items():
        text_clean = text_clean.replace(old_punc, new_punc)

    # 2. Применяем контекстный словарь из config.json (ищем БЕЗ учета регистра)
    replacements = config.get("word_replacements", {})
    for phrase in sorted(replacements.keys(), key=len, reverse=True):
        pattern = re.compile(re.escape(phrase.strip()), re.IGNORECASE)
        # Подставляем точную строку из конфига (с сохранением ЗАГЛАВНЫХ букв-
ударений)
        text_clean = pattern.sub(replacements[phrase], text_clean)

    # 3. Перевод цифр в слова
    def replace_num(match):
        try: return num2words(int(match.group()), lang='ru')
        except: return match.group()
    text_clean = re.sub(r'\d+', replace_num, text_clean)

    # 4. Железный перевод английских слов (игнорируем регистр букв ввода)
    for eng_word, ru_word in ENG_PHONETIC.items():
        pattern = re.compile(r'\b' + re.escape(eng_word) + r'\b',
re.IGNORECASE)
        text_clean = pattern.sub(ru_word, text_clean)

    # Смываем оставшиеся одиночные английские буквы побуквенно для безопасности
    ИИ
    def replace_isolated_lat(match):
        char = match.group().lower()

```

```
    abc =
    {'a': 'a', 'b': 'бэ', 'c': 'цэ', 'd': 'дэ', 'e': 'е', 'f': 'эф', 'g': 'джи', 'h': 'эйч',
    'i': 'ай', 'j': 'джей', 'k': 'кэй', 'l': 'эль', 'm': 'эм', 'n': 'эн', 'o': 'о', 'p': 'пи',
    'q': 'кью', 'r': 'эр', 's': 'эс', 't': 'ти', 'u': 'ю', 'v': 'ви', 'w': 'дабл',
    'x': 'икс', 'y': 'яй', 'z': 'зет'}
    return abc.get(char, char)
    text_clean = re.sub(r'[a-zA-Z]', replace_isolated_lat, text_clean)

    # Убираем лишние пробелы перед отправкой в нейросеть
    text_clean = re.sub(r'\s+', ' ', text_clean)
    return text_clean

class TTSHandler(BaseHTTPRequestHandler):
    def do_POST(self):
        content_length = int(self.headers['Content-Length'])
        post_data =
        json.loads(self.rfile.read(content_length).decode('utf-8'))

        raw_text = post_data.get('text', '')
        base_speaker = post_data.get('speaker', 'baya')
        if base_speaker not in ['aidar', 'baya', 'kseniya', 'eugene',
        'xenia']:
            base_speaker = 'baya'

        config = load_config()
        txt = clean_and_normalize_text(raw_text, config)
        print(f"[ИИ-Синтез] Итоговый текст для Silero: '{txt}' [Голос:
        {base_speaker}]")

        # Модель читает заглавные русские буквы как жесткие ударения
        model.save_wav(text=txt, speaker=base_speaker,
        sample_rate=48000, put_accent=True, put_yo=True)

        self.send_response(200)
        self.send_header('Content-type', 'application/json')
        self.end_headers()
        self.wfile.write(json.dumps({"status":
        "SUCCESS"}).encode('utf-8'))

if __name__ == "__main__":
    server = HTTPServer(('127.0.0.1', 5000), TTSHandler)
    server.serve_forever()
```

From:

<https://wwoss.ru/> - **worldwide open-source software**

Permanent link:

https://wwoss.ru/doku.php?id=local_silero_tts:tts.py

Last update: **2026/07/29 13:21**

