# Отчёт по разработке нейро-ассистента для работы с нормативной документацией

## 1. Тема и описание задачи

**Тема:**  
Разработка нейро-ассистента для работы с нормативной и проектной документацией строительных объектов.

**Описание задачи:**  
Создание полнофункциональной системы, способной:

* Анализировать нормативные документы (СП, ГОСТ, СНиП) и предоставлять точные ответы с цитатами и ссылками на пункты
* Работать с проектной документацией строительных объектов
* Обеспечивать интерактивный диалог через Telegram бот и REST API
* Поддерживать историю диалогов с возможностью суммаризации
* Автоматически обрабатывать документы в различных форматах

## 2. Архитектура системы и обработка данных

**База данных документов:**

* **Форматы:** PDF, TXT, MD
* **Примеры документов:** СП 484.1311500.2020, СП 486.1311500.2020, СП 3.13130.2009 и другие нормативные акты
* **Структура хранения:** Векторная база FAISS с эмбеддингами OpenAI

**Обработка базы данных:**

1. **Загрузка документов:**
   * Для текстовых файлов: чтение с поддержкой multiple encoding (UTF-8, CP1251, Latin1)
   * Для PDF: извлечение текста через PDFMiner с fallback на pdftotext
2. **Предобработка текста:**
   * Автоматическое разделение на абзацы и предложения
   * Сохранение структурной информации о документах
3. **Разделение на фрагменты:**
   * Использование адаптивного чанкера с настраиваемыми параметрами
   * Размер чанка: 2500 символов (конфигурируемо)
   * Перекрытие чанков: 250 символов для сохранения контекста
4. **Векторизация:**
   * Модель эмбеддингов: text-embedding-ada-002
   * Построение FAISS индекса с метаданными документов

**Пример обработки в коде:**

python

def build\_knowledge\_base(self):

"""Построение базы знаний из файлов"""

data\_path = self.config['knowledge\_base']['data\_path']

extensions = self.config['knowledge\_base']['extensions']

all\_documents = []

for ext in extensions:

pattern = f"\*\*/\*.{ext}"

files = list(Path(data\_path).glob(pattern))

for file\_path in files:

if ext == 'pdf':

text = self.processor.read\_pdf\_file(str(file\_path))

else:

text = self.processor.read\_text\_file(str(file\_path))

chunks = self.processor.split\_text(text, file\_path.name)

all\_documents.extend(chunks)

self.knowledge\_base.build\_from\_documents(all\_documents)

## 3. Структура нейро-ассистента

**Архитектурные компоненты:**

1. **Ядро ассистента (assistant.py):**
   * Обработка документов и построение векторной базы
   * Интеграция с OpenAI GPT-4o-mini
   * Поиск релевантных документов и генерация ответов
2. **API сервер (api\_server.py):**
   * REST API на FastAPI для интеграции с внешними системами
   * Эндпоинты для вопросов, суммаризации, управления базой знаний
   * CORS поддержка для веб-интерфейсов
3. **Telegram бот (telegram\_bot.py):**
   * Полнофункциональный интерфейс для пользователей
   * Управление диалогами и историей сообщений
   * Административные функции для управления базой
4. **Менеджер диалогов (dialog\_manager.py):**
   * Сохранение истории сообщений пользователей
   * Суммаризация длинных диалогов
   * Экспорт диалогов в текстовый и JSON форматы
5. **Система тестирования (test\_assistant.py):**
   * Автоматическое тестирование функциональности
   * Генерация отчетов в TXT, CSV и JSON форматах
   * Статистика использования токенов и качества ответов

**Блок-схема обработки запроса:**

text

Пользовательский запрос → Telegram бот / API → Поиск в FAISS →

Формирование контекста → GPT-4o-mini → Форматирование ответа →

Пользователю + сохранение в историю

## 4. Ключевые файлы проекта

1. **Основные модули:**
   * assistant.py - ядро системы с обработкой документов и AI
   * api\_server.py - REST API сервер на FastAPI
   * telegram\_bot.py - Telegram бот с расширенным функционалом
   * dialog\_manager.py - управление диалогами и историей
2. **Конфигурационные файлы:**
   * config.json - настройки системы, моделей, путей
   * templates.json - шаблоны сообщений для интерфейсов
   * prompts.json - промпты для GPT и системные сообщения
3. **Зависимости и запуск:**
   * requirements.txt - зависимости для основной системы
   * requirements\_bot.txt - зависимости для Telegram бота
   * start\_bot.py - скрипт запуска бота с проверкой переменных окружения

## 5. Технические особенности

**Модели и параметры:**

* GPT модель: gpt-4o-mini с температурой 0.1
* Максимальное количество токенов: 1500
* Модель эмбеддингов: text-embedding-ada-002
* Количество релевантных документов для поиска: 5 (конфигурируемо)

**Функциональность:**

* Поддержка multiple encoding для текстовых файлов
* Автоматическое восстановление базы знаний при запуске
* Ограничение истории сообщений (10 на диалог, 50 диалогов на пользователя)
* Административный доступ для перестроения базы знаний
* Система логирования с настраиваемым уровнем детализации

## 6. Выводы

**Достигнутые результаты:**

* Реализован полнофункциональный нейро-ассистент для нормативной документации
* Создана модульная архитектура с разделением ответственности
* Обеспечена поддержка multiple интерфейсов (Telegram, REST API)
* Реализована система управления диалогами с историей и суммаризацией
* Настроена автоматическая обработка документов различных форматов

**Преимущества системы:**

* Высокая модульность и возможность легкого расширения
* Поддержка контекстных диалогов с сохранением истории
* Гибкая конфигурация через JSON файлы
* Комплексная система тестирования и мониторинга
* Административные инструменты для управления системой

**Ограничения:**

* Зависимость от качества исходных документов (OCR для сканированных PDF)
* Ограничения токенов GPT модели для очень сложных запросов
* Необходимость ручной настройки параметров чанкинга для разных типов документов

## 7. План дальнейшей работы

**Ближайшие улучшения:**

1. **Расширение функциональности обработки документов:**
   * Добавление поддержки DOC/DOCX форматов
   * Улучшение обработки таблиц и сложных структур в PDF
   * Реализация веб-интерфейса для загрузки документов
2. **Оптимизация поиска и ответов:**
   * Внедрение гибридного поиска (семантический + ключевые слова)
   * Добавление рейтинга качества ответов пользователями
   * Реализация механизма обратной связи для улучшения ответов
3. **Масштабирование системы:**
   * Добавление поддержки multiple векторных баз для разных категорий документов
   * Реализация кластерного режима для обработки больших объемов запросов
   * Создание системы кэширования частых запросов
4. **Интеграции и расширения:**
   * Добавление веб-интерфейса на Streamlit или Gradio
   * Интеграция с системами документооборота предприятия
   * Поддержка multiple языков для международных нормативных документов
5. **Мониторинг и аналитика:**
   * Реализация панели администратора с метриками использования
   * Система A/B тестирования различных моделей и параметров
   * Детальная аналитика качества ответов и пользовательской удовлетворенности