**ИНСТРУКЦИЯ ПО УСТАНОВКЕ**

**Инструкция по подготовке инфраструктурных компонент**

Для корректной работы цифрового ассистента необходимо подготовить и запустить вспомогательные компоненты, в числе которых:

* Документоориентированная база данных ElasticSearch;
* Key-value база данных Redis (2 инстанса - кеш запросов, бэкенд задач Celery и генерации ответов);
* Распределенная языковая модель с вспомогательной очередью задач Celery;
* Приложение на базе фреймворка FastAPI;
* Веб интерфейс на базе фреймворка Streamlit.

Компоненты цифрового ассистента контейнезированы, запуск полагается на использование docker и docker-compose.

1. Для запуска документоориентированной базы данных подготовлен elasticsearch-configs/docker-compose.yml файл, а также файл elasticsearch-configs/.env.example.
	1. Необходимо скопировать .env.example, создав файл .env, и при необходимости изменить значения версии или пробрасываемых портов (не рекомендуется).
	2. После создания файла .env, в терминале необходимо ввести команду docker compose up --build --detach, после чего база данных будет доступна по URL хоста с портом ${ES\_PORT}

Перед развертыванием других компонент цифрового ассистента, необходимо произвести подготовку данных (парсинг) и загрузку данных в индекс базы данных ElasticSearch (см. Инструкция по подготовке данных).

1. Для развертывания распределенной языковой модели с очередью задач Celery также подготовлен манифест models-serving/docker-compose.yml
	1. Внутри docker-compose.yml необходимо выставить переменные окружения, соответствующие вычислительным ресурсам, на которых будет запущен контейнер:
* CUDA\_VISIBLE\_DEVICES: - номера GPU карт;
* NUM\_OF\_DISTR\_WORKERS: - количество используемых GPU карт;
* MASKING\_TOKENS\_PATH: - путь до маскирующих токенов, если они были получены посредством сборов статистики;
* REDIS\_HOST: - хост контейнера Redis, содержащего бэкенд задач Celery;
* REDIS\_PORT: - порт контейнера Redis, содержащего бэкенд задач Celery;
	1. Также необходимо указать путь до папки (volumes), содержащей веса и токенизатор модели, а также путь подготовленного окружения с предустановленными зависимости из requirements.txt;
	2. После внесения параметров в манифест, в терминале нужно ввести команду docker compose up --build --detach, после чего будет запущен контейнер llama\_llm\_1, в логах которого будет видна задача “run\_completion”.
1. Кеширование запросов происходит в инстанс Redis, для поднятия контейнера, необходимо выполнить команду docker run -v ./MOUNTS\_TTN/redis\_data:/data -p 6381:6379 -d redis:7.2-alpine, где папка ./MOUNTS\_TTN/redis\_data используется для сохранения данных между перезапусками базы данных.
2. Манифест для запуска приложения FastAPI и базы данных Redis, используемой при генерации ответов представлен в docker-compose.yml
	1. В переменных окружения необходимо указать параметры запуска, а именно
* ELASTIC\_HOST: - URL хоста запущенной в п. 1 базы данных ElasticSearch;
* ELASTIC\_PORT: - порт запущенной в п. 1 базы данных ElasticSearch;
* ELASTIC\_INDEX: - индекс ElasticSearch, в который загружены документы;
* REDIS\_HOST: - хост текущего вычислительного узла или “redis”;
* REDIS\_PORT: - 6380, если в поле $REDIS\_HOST указан хост текущего вычислительного узла или 6379 если указан “redis”;
* REDIS\_CACHE\_HOST: - хост вычислительного узла с базой данных Redis (п. 3);
* REDIS\_CACHE\_PORT: 6381
	1. Также необходимо указать путь до сохраненных моделей и файла с эмбеддингами документов в левой части монтируемых volumes или сохранить модели в соответствующей папке ./service\_models\_storage\_lm\_ttn:

 - ./service\_models\_storage\_lm\_ttn:/service\_models\_storage\_lm\_ttn

* 1. Необходимо скопировать .env.example, создав файл .env, и при необходимости заполнить значения пароля и имени пользователя, сконфигурированные при запуске ElasticSearch в п. 1;
	2. После внесения параметров в манифест, в терминале нужно ввести команду docker compose up --build --detach, после чего будут запущены контейнеры с веб-приложением и базой данных.
1. Манифест для запуска веб-интерфейса представлен streamlit-app/docker-compose.yml
	1. Перед запуском внутри манифеста необходимо указать services.app-ui.environment.LLAMA\_API\_HOST параметр, передав в качестве имени хоста адрес вычислительного узла, содержащего приложение FastAPI (п. 4);
	2. Для запуска Web-UI нужно ввести команду docker compose up --build --detach, которая поднимет контейнер с интерфейсом по адресу <current\_node\_hostname>:8501

**Инструкция по подготовке данных, генерации и дообучению**

1. Подготовка документов (парсинг)
2. Все документы, подготавливаемые к загрузке в базу данных, нужно поместить в отдельную папку и указать путь до нее в файле documents\_processing/config.yaml в переменной docs\_dir\_path. Отдельно необходимо создать пустую папку, в которую будут выгружены результаты парсинга, путь до созданной папки указывается в том же файле в переменной docs\_save\_path.
3. Для запуска процесса подготовки данных необходимо набрать команду python documents\_processing/examples/example\_parser.py.
4. Отдельно необходимо подготовить документы для парсинга таблиц, для этого все документы, в которых содержатся таблицы для парсинга должны быть помещены в отдельную папку, путь до которой должен быть указан в поле tables\_dir\_path конфигурационного файла, аналогично п. a, папка для сохранения результатов должна быть указана в поле tables\_save\_path.
5. Все необходимые документы после парсинга документов, то есть файлы в формате .parquet , которые получаются после запуска файла, documents\_processing/src/parsing/parser.py, должны быть помещены в отдельную папку (параметр tables\_input\_parquet\_doc).. Документы п. a должны быть переведены в формат .pdf и помещены в отдельную папку, путь до нее должен указываться в параметре tables\_input\_pdf\_file.
6. Парсинг таблиц производится командой python documents\_processing/examples/table\_example.py.
7. Для создания словаря соответствий между названиями документов и его основными заголовками (содержания), все соответствующие документы в формате .parquet должны быть помещены в отдельную папку (путь до нее должен быть указан в конфигурационном файле под ключом mapping\_dir\_path). Путь до папки, в которой будут сохранены результаты маппинга, указывается в параметре mapping\_save\_path.
8. Запуск Python скрипта производится с помощью команды python documents\_processing/examples/mapping\_example.py.
9. Загрузка документов в базу данных ElasticSearch
	1. В конфигурационном файле documents\_processing/config.yaml необходимо указать следующие параметры:
* raw\_docs\_dir: - путь до исходных документов, которые были обработаны в п. 1;
* parsed\_docs\_dir: - путь до обработанных и сохраненных в п.1 .parquet файлов
* es\_host: - полный URL ElasticSearch контейнера ("http://<IP хоста вычислительной ноды>:${ES\_PORT}")
* es\_index\_name: - наименование создаваемого / переписываемого индекса ElasticSearch, в который будут выгружены документы (соответствующий $ELASTIC\_INDEX параметру запуска приложения FastAPI)
	1. Для запуска процесса загрузки данных необходимо запустить скрипт посредством ввода команды python documents\_processing/examples/docs\_searcher\_example.py
1. Генерация синтетических примеров для дообучения моделей
	1. В файле syn\_generation/config.yaml указать следующие параметры:
		1. docs\_df\_path: - путь до датасета с обработанными текстами
		2. gen\_df\_path: - путь по которому необходимо сохранить сгенерированные вопросно-ответные пары
		3. process\_col\_name: - название колонки с данными
		4. path\_to\_model: - путь до модели
	2. Для запуска процесса генерации необходимо запустить скрипт посредством ввода команды python syn\_generation/src/generation.py
	3. Для запуска процесса пост-фильтрации примеров необходимо запустить скрипт python syn\_generation/src/filtering.py
	4. Для обеспечения более высокого качества обучения модели рекомендуется произвести проверку сгенерированных примеров
2. Дообучение модели
	1. Вносим необходимые значения параметров в файл fine-tuning/src/scripts/run\_training.sh
	2. Запускаем дообучение путем вызова скрипта python fine-tuning/src/train\_seq2seq\_qa.py