в офлайн-режиме. Service Worker помогает ускорить последующие загрузки, кэшируя статические файлы и данные, что делает приложение более отзывчивым.

После установки на устройство PWA ведет себя как нативное приложение. Оно может запускаться из иконки на домашнем экране и отображаться в режиме полного экрана, без видимых элементов браузера, таких как адресная строка или кнопки навигации. Это создает более интегрированный и естественный опыт использования для пользователя.

Первый запуск PWA закладывает основу для его последующего использования, обеспечивая установку, кэширование необходимых ресурсов и интеграцию с устройством пользователя для улучшенного взаимодействия.

Список источников

1. Джексон М. Архитектура современных веб-приложений. Москва : ДМК Пресс, 2017. 400 с.
2. Дубейковский В. И. Практика функционального моделирования с AllFusion Process Modeler. Москва : ДИАЛОГ-МИФИ, 2021. 464 с.

УДК 004.655

ИСПОЛЬЗОВАНИЕ ВЕКТОРНЫХ БАЗ ДАННЫХ ДЛЯ СТРУКТУРИЗАЦИИ ТЕХНИЧЕСКИХ ЗАДАНИЙ С ПОМОЩЬЮ БОЛЬШИХ ЯЗЫКОВЫХ МОДЕЛЕЙ

Е.М. Веселов[1], *В.А. Тимченко*[2]
Научный руководитель В.В. Грибова
Владивостокский государственный университет
Институт автоматики и процессов управления Дальневосточного отделения Российской академии наук, Владивосток, Россия,
[1]*evgeniyvsl@vk.com,* [2]*vadim@dvo.ru*

В текущей действительности одним из важных факторов для развития и масштабирования производственной деятельности является цифровизация. Цифровизация любой сферы хозяйственной деятельности способствует повышению качества ведения хозяйственного учёта и оптимизации издержек. Однако, с увеличением темпов развития, возрастает и потребность в повышении скорости обработки

данных. Векторные базы данных (далее векторные БД) – современное цифровое решение, которое может как повысить скорость работы с данными, так и открыть возможность для удобной интеграции с другими современными технологиями, такими как большие языковые модели или Large Language Models (далее LLM). В рамках данной работы будет рассмотрен аспект, в котором векторные БД могут быть задействованы в цифровом решении по извлечению и структуризации информации из технических заданий (далее ТЗ) на выполнение технологических операций по лазерной аддитивной обработке материалов.

К ТЗ данного типа применяется ГОСТ 15.016-2016 «Система разработки и постановки продукции на производство. Техническое задание. Требование к содержанию и оформлению». Проблема в том, что данный ГОСТ не предъявляет достаточное количество строгих требований. Обязательные моменты для соблюдения: информация в ТЗ должна соответствовать законодательству РФ, а также предусматривать выполнение других нормативных актов, относящихся к продукции, являющейся объектом текущего ТЗ [1]. Наличие прочей информации носит рекомендательный характер, из-за чего ТЗ сильно различаются как по структуре, так и по наполнению. Это увеличивает время настройки режимов работы оборудования для выполнения операции лазерной наплавки в соответствии с содержащейся в ТЗ информации. Сочетание векторных БД и LLM могут помочь в решении проблемы структуризации и формализации неоднородных ТЗ, что позволит упростить и автоматизировать процесс настройки оборудования и тем самым повысить эффективность работы предприятия. Для рассмотрения данного вопроса стоит углубиться в технический аспект.

Векторные БД – это тип баз данных, который хранит информацию в виде векторизованных значений. Сами вектора – это математические объекты, характеризующиеся величиной и направлением. Вектор может быть представлен в пространстве в виде упорядоченного набора чисел, называемого координатами [2, с. 156]. Современные технологии позволяют преобразовать любой необходимый для обработки формат данных (числовой, текстовый, графический, аудио-формат) в векторное представление. Данные, преобразован-

ные в векторный формат, называются эмбеддингами («embeddings»). Упорядоченное хранение эмбеддингов в векторном пространстве – задача векторных БД. Плюс подхода работы с текстами (данными) в виде эмбеддингов, а не в изначальном (текстовом) виде, заключается в возможности сохранения семантической связи между словами в том виде, в котором они будут понятны для обрабатывающей их LLM.

Понимание семантической близости между словами в представлении, доступном для машинной обработки, достигается за счёт возможности рассчитать расстояние между двумя векторами. Модель для создания эмбеддингов преобразует поступающие на вход тексты в точки, которые размещаются в векторной БД. Соответственно, если узнать расстояние между точками в векторном пространстве, можно понять, в каком контексте слово может считаться уместным. Чем меньше расстояние между точками, тем больше слова «подходят» друг другу. На этой идее основан один из самых популярных методов векторизации слов – Word2Vec. В упрощённой форме процесс работы данного метода преобразования текстов в эмбеддинги можно описать следующим образом: модель анализирует статистику совместной встречаемости слов, определяя, сколько раз каждое слово встречается рядом с другими словами. На основе этой информации можно построить вектор частоты совместной встречаемости слова с другими словами в текстовом окне заданной длины. Если два слова встречаются с одними и теми же словами с примерно одинаковой частотой, эти слова считаются семантически похожими [3, с. 63]. Если два слова практически не встречаются рядом друг с другом, то есть, расстояние между ними в векторном пространстве является большим, они считаются семантически различными. Данный принцип активно используется, например, в технологиях, лежащих в основе «подсказывания» следующего слова при наборе текста пользователем. Визуализация связи слов в векторном пространстве представлена на рис. 1.

Векторные БД, за счёт индексации поступающих на вход векторов, позволяют осуществлять быстрый поиск даже среди большого количества записей. Стоит иметь в виду, что векторные БД – это NoSql решение, поэтому невозможно получить строгую транзак-

ционность и соблюдение принципов ACID, что может быть важно в сферах со строгой регламентацией поступающих в БД данных: банковские операции, документооборот, складской учёт и т. д. В рамках затрагиваемой в статье тематики, возможность поиска в векторной БД без указания однозначных ключевых значений становится плюсом при обучении языковых моделей. Во время обучения LLM учится предсказывать следующее слово в предложении, учитывая предшествующие слова. Обработав достаточное количество текстов, модель начинает лучше «понимать», какое слово будет уместным в текущем контексте. Это помогает модели строить правильные предложения и избегать ситуаций с двойственным значением слов.

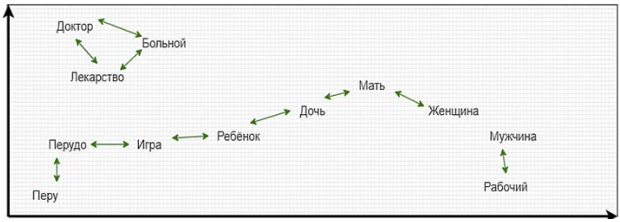

Рис. 1. Семантическая связь слов в векторном пространстве

Описанный выше принцип построения предложений даёт понимание того, как языковая модель работает с поступающим от пользователя запросом. Однако, полноценное дообучение модели или же Fine-tuning – операция, требующая большого количества вычислительных ресурсов и набора обучающих данных, исчисляемого текстами из десятков или сотен тысяч слов. Для решения задачи структуризации технических заданий есть иной механизм посвящения LLM в контекст собственных локальных документов – RAG (Retrieval Augmented Generation) или «Генерация, дополненная результатами поиска». Название отражает принцип работы данной технологии: вместе с запросом от пользователя, модель получает на вход контекст, который хранится в виде эмбеддингов в векторной базе данных. Это не является полноценным дообучением, но позволяет модели ориентироваться среди гораздо большего количества информации, чем можно подать на вход при классическом запросе [4, p. 4–5]. Благодаря описанным ранее принципам работы векторов

и векторных БД, операция поиска информации в предоставленном контексте занимает мало времени и LLM может предоставить пользователю ответ не только на основании общих данных, усвоенных при обучении, но и при помощи локальной информации, предварительно размещённой в векторной БД.

Тестирование данного подхода проводилось при помощи векторной БД Chroma и локальной версии готовой языковой модели Llama2 с 7 млрд. параметров. Chroma была выбрана в силу наличия интеграции с Langchain (платформа для взаимодействия с LLM) и подробной документации. Для преобразования текста в эмбеддинги использовалась функция OllamaEmbeddings, задействующая вычислительный механизм Llama2 и также имеющая интеграцию с Langchain. Перед загрузкой документов в векторную БД тексты были разбиты на части – чанки (англ. – chunks). Длина чанков составила 800 символов с пересечением между соседними чанками длиной в 80 символов. Разделение на чанки оптимизирует процесс преобразования текста в эмбеддинги и загрузки их в векторную БД. Chroma позволяет присваивать каждому элементу, загружаемому в неё, отдельное обозначение – метаданные. Перед загрузкой каждому чанку были присвоены метаданные, содержащие название загружаемого файла. Это позволяет различать источник чанков при дальнейшей работе с БД и проводить следующие операции:

– определять, что подаваемый на вход документ был загружен ранее и не производить деление на чанки, преобразование в эмбеддинги и загрузку в БД;

– проверять наличие документа в векторной БД во время запроса пользователя. Если документ, который запрашивает пользователь, не содержится в базе, запрос не передаётся к LLM, что сокращает время ответа, т.к. поиск элемента в векторной БД ощутимо быстрее, чем обработка запроса локальной LLM;

– удалять данные из векторной БД. Удаление происходит при помощи указания уникального идентификатора элемента БД. Так как текст разбивается на чанки и один изначальный документ превращается во множество эмбеддингов, операция полного удаления всех связанных с нужным файлом эмбеддингов становится затруднительной. Добавленные метаданные позволяют найти все эмбед-

динги, относящиеся к запрошенному документу, получить их уникальные идентификаторы и удалить из БД в операции цикла.

Загрузка документов в векторную БД производилась на ПК с 16 ГБ оперативной памяти, процессором Intel Core i5-1135G7 и твердотельным накопителем памяти Samsung MZVLB512HBJQ. На обозначенной конфигурации время загрузки pdf-документа размером 3,5 МБ составила 6 минут и 48 секунд. Заметим, что указанная конфигурация не является высокопроизводительной и использование более мощных вычислительных ресурсов позволит добиться ощутимого ускорения времени загрузки. После успешной загрузки документа в базу данных, при необходимости, можно получать информацию о её содержимом в формате JSON. Древовидная структура JSON позволяет извлекать и обрабатывать всю необходимую информацию в необходимом для исследования виде. Например, для вывода списка документов, хранящихся в базе данных не в виде нечитаемого для человека набора чисел, а в привычном текстовом представлении.

После наполнения векторной БД, размещённые в базе эмбеддинги можно передать используемой LLM, указав путь к локальной базе данных и задав контекст, которому модель должна следовать во время генерации ответа (рис. 2).

```
CHROMA_PATH = "chroma"
LLM_model = "llama2"

PROMPT_TEMPLATE = """
Пользователь предоставит название файла, содержащего техническое задание.
Задача заключается в том, чтобы внимательно изучить данный файл и предоставить следующую информацию:

1) Цель выполнения технологической операции;
2) Используемый тип порошкового питателя;
3) Конфигурация лазерной оптической головки.

Ответы должны быть представлены исключительно в виде тезисов.
В случае, если необходимая информация отсутствует, укажите "не найден" напротив соответствующего пункта.
Не предоставляйте никакой дополнительной информации, ограничиваясь только указанными выше пунктами, которые выделены символами

{context}
"""
```

Рис. 2. Подготовка контекста для ответов LLM

Следует принять во внимание, что во время работы с LLM эмбеддинги (данные), найденные для ответа на запрос пользователя, будут преобразованы обратно в текстовое представление. При этом важно применить для обратного преобразования ту же функцию, которая была использована для векторизации изначального текста.

Использование другой функции может привести к искажению информации и ухудшению качества полученного ответа.

В результате проведённого исследования была развёрнута локальная векторная БД, содержащая в себе ТЗ на операцию лазерной наплавки в векторизованном виде. Также был реализован функционал по корректному преобразованию текста в векторное представление и проверки на то, добавлялся ли ранее данный документ в векторную БД. Дальнейшие работы по теме данного исследования будут сконцентрированы на отлаживании функционала по получению ТЗ из векторных БД. Ответы должны содержать в себе структурированные ТЗ, которые запросил пользователь. Структуризация будет проводиться на основе подготовленного шаблона, при помощи локальной LLM с открытым исходным кодом или проприетарной модели, предоставляющей свой функционал посредством API.

Работа выполнена в рамках государственного задания ИАПУ ДВО РАН (тема № FWFW-2025-0004).

Список источников

1. ГОСТ 15.016–2016. Система разработки и постановки продукции на производство. Техническое задание. Требования к содержанию и оформлению : межгосударственный стандарт : издание официальное : принят Межгосударственным советом по стандартизации, метрологии и сертификации (протокол от 25 октября 2016 года № 92-П) : введен впервые : дата введения 2017-09-01. Переизд. Москва : Стандартинформ, 2020. III, 27 с.
2. Кравченко Ю. А., Мансур А. М., Мохаммад Ж. Х. Векторизация текста с использованием методов интеллектуального анализа данных // Известия ЮФУ. Технические науки. 2021. № 2. С. 154–166.
3. Ярушкина Н. Г., Мошкин В. С., Константинов А. А. Применение языковых моделей word2vec и bert в задаче сентимент-анализа текстовых сообщений социальных сетей // Автоматизация процессов управления. 2020. № 3. С. 60–69.
4. Retrieval-Augmented Generation for Large Language Models: A Survey / Y. Gao, Y. Xiong, X. Gao [et al.] // arXiv : сайт / Cornell University. URL: arxiv.org/abs/2312.10997 (дата обращения: 09.04.2025).