МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО
ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ
РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ИМЕНИ В.Ф. УТКИНА

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В ПРИКЛАДНЫХ ИССЛЕДОВАНИЯХ

Межвузовский сборник научных трудов

Рязань 2023

УДК 004

Информационные технологии в прикладных исследованиях, межвузовский сборник научных трудов. – Рязань: ИП Коняхин А.В. (Book Jet), 2023. – 246 с.

ISBN 978-5-907568-65-5

Публикуются статьи о проблемах использования информационных технологий в науке, промышленности и образовании.

Сборник рассчитан на научных сотрудников, преподавателей, аспирантов и студентов высших учебных заведений.

ISBN 978-5-907568-65-5

СОДЕРЖАНИЕ

УДК 004.77

А.В. АГАПОВ
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ПАРСИНГ ИНТЕРНЕТ-РЕСУРСОВ

В статье рассматриваются цели и способы парсинга Интернет-ресурсов, преимущества инструментов парсинга и пример использования библиотеки Jsoup.

Парсинг (анализ синтаксической структуры) – это процесс анализа текстов или других данных, в котором программа разбирает их на составляющие части, определяет их роль и взаимосвязь. Например, парсинг может использоваться для анализа HTML-кода веб-страницы, чтобы извлечь информацию о содержании, структуре и свойствах элементов на странице. Парсинг может быть выполнен как вручную, так и автоматически с помощью специальных программных инструментов, называемых парсерами.

Парсеры сайтов – это инструменты, которые позволяют автоматически собирать данные с веб-страниц и использовать их для различных целей, например, для анализа рынка, мониторинга конкурентов, создания баз данных или для построения различных сервисов и приложений. В настоящее время парсеры сайтов являются необходимым инструментом для многих компаний и предпринимателей, которые хотят получить доступ к большому объему данных, но не имеют времени на ручной сбор информации.

Парсер может быть полезен программисту во многих ситуациях.

1. Анализ и разбор кода. Парсер может помочь программисту понять структуру и синтаксис кода, извлечь информацию о переменных, функциях, классах и других элементах программы.

2. Работа с данными. Парсер может помочь программисту извлечь нужную информацию из различных источников данных, таких как файлы различных форматов, базы данных и т.д.

3. Автоматизация задач. Парсер может использоваться для автоматизации повторяющихся задач, таких как обработка данных и генерация отчетов.

4. Разработка инструментов. Парсер может быть использован для разработки инструментов, которые помогут программисту ускорить и упростить разработку программного обеспечения.

5. Тестирование программного обеспечения. Парсер может использоваться для тестирования программного обеспечения,

например, для проверки правильности работы парсера или для тестирования взаимодействия с другими системами.

6. Разработка искусственного интеллекта. Парсер может использоваться для обработки и анализа больших объемов данных, которые используются для обучения искусственного интеллекта.

В целом, парсер может быть полезен программисту во многих ситуациях, где требуется обработка и анализ данных.

Основные этапы парсинга.

1. Определение формата данных. Для успешного парсинга необходимо знать формат данных, в котором они представлены, например, XML, JSON, CSV или текстовый файл.

2. Выбор инструментов. Для парсинга данных можно использовать различные инструменты, в том числе языки программирования (Python, Java, PHP), библиотеки (BeautifulSoup, lxml), утилиты командной строки (grep, awk), онлайн-сервисы и т.д.

3. Написание кода. После выбора инструментов необходимо написать код, который будет осуществлять парсинг данных. Код должен содержать инструкции и методы, которые позволяют получить нужную информацию из данных.

4. Тестирование. После написания кода необходимо провести тестирование парсера, чтобы убедиться в его правильной работе. В процессе тестирования необходимо проверить различные случаи использования, например, наличие ошибок и неожиданных данных.

5. Улучшение и оптимизация. После тестирования парсер может быть доработан для улучшения скорости и точности работы.

Лучше всего использовать язык программирования Python для создания парсеров, т.к. он обладает самыми мощными и функциональными библиотеками.

Самым лучшим инструментом считается фреймворк Scrapy. Он позволяет создавать мощные и гибкие парсеры, которые могут обрабатывать большие объемы данных. Scrapy поддерживает многопоточность, асинхронность и распределенный парсинг, что позволяет ускорить процесс сбора данных.

Однако в данной статье хочется уделить внимание другому языку программирования Java, у которого есть множество инструментов для парсинга данных. Рассмотрим наиболее популярные из них.

1. JSoup – это одна из самых популярных библиотек для парсинга HTML и XML документов. Она обладает простым и удобным интерфейсом, позволяет выбирать элементы по CSS-селекторам, имеет встроенный парсер HTML и XML документов. Ее главным

преимуществом является простота использования и высокая скорость работы. Однако она не поддерживает некоторые расширенные функции парсинга, такие как XPath.

2. Selenium – это инструмент для автоматизации тестирования веб-приложений, который также может использоваться для парсинга веб-страниц. Selenium имитирует браузер и выполняет JavaScript на странице, что позволяет получить более точный результат парсинга. Однако, его использование может быть затруднено из-за сложности настройки и обучения, а также необходимости установки дополнительных драйверов.

3. Apache HttpComponents – это библиотека для отправки HTTP запросов и получения HTTP ответов. Она может использоваться для получения HTML и XML документов из Интернета. Главным преимуществом является низкий уровень абстракции и возможность полного контроля над запросами и ответами. Однако она не имеет возможности выбирать элементы по CSS-селекторам и не поддерживает JavaScript.

Рассмотрим подробнее библиотеку JSoup и напишем свой собственный парсер. Собирать данные будем из HTML-кода веб-страницы сервиса подбора слов Яндекса, который предназначен для анализа частотности запросов в поисковой системе Яндекс.

С помощью этого сервиса можно узнать, сколько раз определенный запрос был введен пользователями в поисковую строку Яндекса за определенный период времени. Для этого нужно ввести запрос в поисковую строку сервиса и указать период анализа (неделя, месяц, год и т.д.).

Допустим, мы хотим узнать, сколько раз пользователи вводят запрос «Погода в Рязани» (рисунок 1).

Что искали со словом «погода в рязани» — 457 616 показов в месяц

Статистика по словам	Показов в месяц
погода +в рязани	285 506
погода +в рязани +на неделю	68 940
погода +в рязани +на день	43 202
погода +в рязани +на 14	36 216
прогноз погоды рязань	28 944
погода +в рязани +на 14 дне	26 798
погода +в рязани +на 14 дней	26 798
погода +в рязани +на месяц	15 388
погода точный рязань	14 563
погода рязань 2023	11 992

Рисунок 1 – Статистика запроса

Видим 457 616 показов в месяц и самый популярный запрос «погода +в рязани», у которого 285 506 показов в месяц. Оператор «+» ставят перед стоп-словами и предлогами для точного их включения. По умолчанию они не учитываются.

Чтобы спарсить первые десять результатов запросов, для начала скачаем библиотеку Jsoup с официального сайта и в среде разработки подключаем библиотеку. Необходимо создать объект, в который будем сохранять HTML-кода страницы. Сделать это можно двумя способами: загрузкой с URL или загрузкой с файла. В нашем случае загрузка будет с файла. Ниже представлена функция getPage(), возвращающая код страницы, который будет использоваться для анализа.

```
private static Document getPage() throws IOException {
    String filename =
"D:\\IdeaProjects\\Parser\\WordStat.txt";
    File input = new File(filename);
    Document page = Jsoup.parse(input, "UTF-8");
    return page;
}
```

Получение нужных элементов страницы с помощью метода select():

```
//получаем страницу по запросу
Document page = getPage();
//получаем статистику по словам
Elements phraseBasic = page.select("a[class=b-link b-
phrase-link__link]");
//получаем количество показов в месяц
Elements tableCounts = page.select("td[class=b-word-
statistics__td b-word-statistics__td-number]");
//перевод в массив строк
String [] splitedCount = tableCounts.text().split(" ");
//перевод в массив чисел
int[] splitedPhrase = tableCounts.split(" ");
```

Далее можно вывести результаты в консоль, однако данный метод не удобен и лучше всего перенести их в csv-файл или xls(xlsx)-файл. Для этого потребуется установить ещё одну библиотеку: Apache POI. Она представляет собой API, который позволяет использовать файлы MS Office в Java приложениях.

```
    //создаем новую excel книгу
Workbook wb = new XSSFWorkbook();
//создание листа 1
XSSFSheet sheet1 = (XSSFSheet)
wb.createSheet("mySheet1");
//создаем стили ячеек
```

```
CellStyle style = wb.createCellStyle();
style.setAlignment(HorizontalAlignment.CENTER);
style.setVerticalAlignment(VerticalAlignment.CENTER);
     //объявление строки
XSSFRow row;
     //создали 1 строку
row = sheet1.createRow(0);
     //создали 1 колонку
Cell cell0 = row.createCell(0);
     //присвоили стиль к 1 колонке
cell0.setCellStyle(style);
Cell cell1 = row.createCell(1);
cell1.setCellStyle(style);
cell0.setCellValue("Ключевое слово");
cell1.setCellValue("Показов в месяц");
     //цикл для заполнения таблицы
for (int i = 0; i < 10; i++) {
    row = sheet1.createRow(i+1);
    for (int j = 0; j < 2; j++) {
        Cell cell = row.createCell(j);
        cell.setCellStyle(style);
        switch (j) {
            case 0 ->
cell.setCellValue(splitedPhrase[i]);
            case 1 ->
cell.setCellValue(splitedCount[i]);
        }
    }
}
//авторасширение колонок под размер содержимого
for (int i = 0; i < 10; i++) {
    sheet1.autoSizeColumn(i);
}
//поток для создания файла
FileOutputStream fos = new FileOutputStream("my2.xlsx");
     //запись в поток
wb.write(fos);
     //закрыть файловый поток
fos.close();
```

В результате получим xlsx-файл (рисунок 2), в который быстро и автоматически перенеслась вся необходимая нам информация с сайта по запросу. Можно задавать сколько угодно запросов, и они все будут сохраняться в данном файле.

my2.xlsx

	A	B	C	D	E
1	Ключевое слово	Показов в месяц			
2	погода +в рязани	285506			
3	погода +в рязани +на неделю	68940			
4	погода +в рязани +на день	43202			
5	погода +в рязани +на 14	36216			
6	прогноз погоды рязань	28944			
7	погода +в рязани +на 14 дне	26798			
8	погода +в рязани +на 14 дней	26798			
9	погода +в рязани +на месяц	15388			
10	погода точный рязань	14563			
11	погода рязань 2023	11992			
12					
13					

Рисунок 2 – Созданный парсером xlsx-файл

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Парсинг HTML с помощью jsoup. – Текст: электронный. – URL: https://javarush.com/groups/posts/2007-legkiy-parsing-html-s-pomojshjhju-jsoup (дата обращения: 15.04.2023).

2. Apache POI – Краткое руководство Текст: электронный. – URL: https://coderlessons.com/tutorials/java-tekhnologii/izuchite-apache-poi/apache-poi-kratkoe-rukovodstvo (дата обращения: 15.04.2023).

3. Яндекс.Метрика: инструкция по веб-аналитике – Бесплатное электронное издание // Коллектив авторов под редакцией Е. Илюшкиной и А. Косолобова. – ООО «Ингейт Реклама», 2015. – 49 с.

УДК 004.65

В.А. БАРАНОВ
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ПРОБЛЕМЫ И ПЕРСПЕКТИВЫ РАЗВИТИЯ ИНТЕРНЕТА ВЕЩЕЙ (IOT)

Рассматривается технология интернета вещей (IOT), ее развитие, проблемы и перспективы.

Новые технологии в производстве играют важную роль в повышении производительности. Они позволяют автоматизировать

многие процессы, уменьшить время потребления ресурсов, а также повысить точность работы. К примеру, роботизированное производство позволяет сократить время на производство и уменьшить количество отходов. Использование Интернета Вещей (IoT) позволяет контролировать использование ресурсов, а также позволяет быстро реагировать на сбои в производстве. Использование искусственного интеллекта (AI) помогает улучшить контроль качества, оптимизировать процессы производства и сократить потребление ресурсов. 3D-печать промышленных товаров позволяет сократить время на производство и уменьшить расходы на транспортировку. Новые технологии в производстве также дают возможность производителям получить новые данные и с помощью анализа этих данных оптимизировать свои процессы. Все эти технологии позволяют улучшить производительность, снизить издержки, увеличить прибыльность и улучшить конкурентоспособность.

Интернет вещей (IoT, Internet of Things) представляет собой сеть устройств, связанных друг с другом, которые могут обмениваться информацией через Интернет. Эти устройства могут быть разного рода: от домашних умных приборов, таких как холодильники и кондиционеры, до бизнес-систем управления запасами и устройств отслеживания рабочих процессов.

Использование Интернета вещей может иметь множество полезных применений.

1. Умный дом: Устройства IoT могут управлять светом, температурой и другими элементами домашней автоматизации, что позволяет улучшать комфортность и энергосбережение.

2. Город: Включение устройств IoT в инфраструктуру города может улучшить управление городской средой, от управления уличным освещением до управления транспортом и отходами.

3. Промышленность: Устройства IoT могут собирать данные о производственных процессах, чтобы помочь оптимизировать производственные процессы и улучшить качество продукции.

4. Здравоохранение: IoT может помочь в разработке систем для мониторинга и управления здоровьем, например, систем для контроля кровяного давления и пульса.

5. Сельское хозяйство: IoT может помочь в контроле производства продуктов питания, например, для определения оптимального времени для полива, увеличения урожайности и уменьшения затрат.

Интернет вещей (IoT) – это набор технологий и принципов, позволяющих взаимодействовать и обмениваться данными между

устройствами в реальном времени без участия пользователя. Развитие этой технологии началось в прошлом веке, когда компьютеры только появились.

Первым устройством, которое можно считать прообразом IoT, был Trojan Room Coffee Pot – автоматическая кофеварка, установленная в 1991 году в лаборатории Кембриджского университета. Когда кофеварка закипала, она отправляла сообщение на компьютерные мониторы в лаборатории, предупреждая о том, что кофе готов.

В 2008 году появилась первая версия беспроводной технологии ZigBee, позволяющей устройствам обмениваться характеристиками и данными на коротких расстояниях. Это был важный шаг к созданию IoT, поскольку технология позволяла создавать простые и недорогие устройства, способные взаимодействовать друг с другом.

В 2010 году IBM представила концепцию «умного города», которая включала в себя систему управления транспортным потоком, энергоснабжением и общественными услугами, все управлялось с помощью IoT устройств.

В настоящее время IoT активно используется в таких сферах, как медицина, транспорт, городское хозяйство, производство и домашнее хозяйство. Каждый день появляются новые устройства, которые используют IoT, например, устройства умного дома, термостаты, беспилотные автомобили и даже умные протезы.

Сегодня IoT помогает компаниям и частным лицам повысить свою эффективность и экономить средства, улучшая повседневную жизнь. Однако, вместе с возможностями, этой технологии существует и опасность нарушения конфиденциальности и безопасности данных. Поэтому, создание современных систем IoT должно уделять особое внимание защите информации и продумыванию мер по ее сохранности.

Следует отметить, что IoT создает огромное количество данных, которые нужно обрабатывать и анализировать. Поэтому ключевым фактором становится развитие искусственного интеллекта, что бы уметь эффективно управлять всеми этими данными.

Достоинства Интернет вещей.

1. Улучшенное управление энергопотреблением.

Централизованный мониторинг потребления энергии может помочь снизить издержки на электроэнергию. Например, "умный дом" сможет оптимизировать потребление энергии и позволить снизить затраты.

2. Улучшенное управление ресурсами.

IoT может использоваться для контроля использования ресурсов, таких как вода, и предотвращения их расточительного использования. В результате, можно ожидать экономии и более устойчивого будущего для человечества.

3. Лучшее здравоохранение.

IoT может применяться для мониторинга здоровья пациентов в режиме реального времени и оповещать врачей об изменениях в состоянии пациента. Это может помочь предотвратить многие болезни или помочь в ранней идентификации.

4. Удобство и автоматизация.

IoT может повысить комфорт жизни и улучшить ежедневную рутину. Например, компьютер может запустить стиральную машину, когда вы собираетесь уйти из дома, позволяя вам избежать забывания об этом.

5. Улучшенная безопасность.

IoT системы могут помочь защитить владения от взломов и несанкционированного доступа. Например, системы безопасности можно использовать для защиты домов, установив видеонаблюдение, контроль доступа, отслеживание движения.

Недостатки Интернет вещей.

1. Ограничения пропускной способности.

В IoT-сетях возникают проблемы с пропускной способностью, что может снизить быстродействие многих устройств, нередко приводя к низким показателям качества обслуживания.

2. Ограничения безопасности.

IoT-устройства могут подвергаться все большему числу кибератак, включая атаки DDoS, фишинг и многие другие. Кроме того, данные, которые генерируются IoT-устройства, часто содержат личную информацию владельцев, что может быть украдено и использовано нежелательным образом.

3. Охват сети.

Некоторые устройства IoT могут не подключаться к интернету на удалении, если там плохой или отсутствует сигнал, что ограничивает использование.

4. Требования к питанию.

Многие устройства IoT требуют постоянного питания, что может быть проблемой.

В целом, Интернет вещей – это перспективная технология, которая позволяет значительно улучшить нашу повседневную жизнь, повысить эффективность и сделать наш мир более удобным и безопасным. Однако, чтобы успешно реализовать эту технологию,

необходимо решить ряд серьезных задач, связанных с обеспечением конфиденциальности и безопасности.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Антти Суомалайнен. Интернет вещей: видео, аудио, коммутация. – М.: ДМК Пресс, 2019. – 120 с.

2. Проблемы безопасности Интернета вещей: учебное пособие. – М.: Мир науки, 2021.

УДК 004.65

В.А. БАРАНОВ
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ПЕРСПЕКТИВЫ СИСТЕМ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ

Рассматриваются перспективы систем автоматизированного проектирования в оптимизации производства.

Автоматизация процессов проектирования – это использование компьютерных технологий и программных средств для упрощения и ускорения процесса разработки проектов. Задачи проектирования включают в себя анализ требований, проектирование, моделирование, тестирование и документирование проектов.

Автоматизация процессов проектирования может быть применена в разных областях:

1. Создание 3D-моделей деталей и сборок.
2. Генерация чертежей и документации.
3. Анализ напряжений и деформаций объектов.
4. Определение массы и центров масс объектов.
5. Разработка и оптимизация траекторий движения роботов и машин.
6. Симуляция и виртуальное тестирование работы объектов.
7. Проектирование электрических схем и печатных плат.
8. Разработка исходных кодов для программирования микроконтроллеров.
9. Создание и анализ топологии электрических и механических систем.
10. Интеграция с другими программными средствами и обмен данными с ними.

Для автоматизации процессов проектирования используются различные программные средства, такие как САПР (системы автоматизированного проектирования), CAD (системы компьютерного проектирования), CAM (системы компьютерного проектирования для производства) и другие. Эти программы позволяют разработчикам быстро создавать и изменять проекты, проверять их на стадии проектирования и сокращать процесс разработки до нескольких дней вместо нескольких недель или месяцев.

САПР – это система автоматизированного проектирования. С помощью САПР можно создавать различные проекты – от электрических схем и печатных плат до больших промышленных сооружений и зданий. САПР включает в себя различные программы и инструменты для проектирования, моделирования, расчетов и тестирования проектов, что позволяет повысить производительность и качество процесса проектирования. САПР является неотъемлемой частью современной индустрии и позволяет существенно сократить время и затраты на проектирование.

Как и с любой другой технологией, нет единого "лучшего" САПР (системы автоматизированного проектирования), так как каждая из них имеет свои преимущества и недостатки и может быть более подходящей для определенных задач или отраслей. Некоторые из наиболее распространенных САПР включают:

1. AutoCAD – одна из самых известных САПР, используемых для 2D и 3D-моделирования, проектирования и документирования.

2. SolidWorks – мощная САПР, специализирующаяся на механическом проектировании и 3D-моделировании.

3. CATIA – широко используемая САПР для проектирования и разработки промышленных и автомобильных изделий, а также для авиационной и космической отраслей.

4. Revit – популярная САПР для архитектурного проектирования, которая включает инструменты для создания 3D-моделей зданий и конструкций.

5. SketchUp – интуитивная САПР для дизайна и архитектурного моделирования, которая включает мощный набор инструментов для создания 2D- и 3D-моделей.

6. Fusion 360 – интегрированная САПР для 3D-моделирования, проектирования и CAM, которая позволяет единым проектам легко доступным в облачном хранилище.

Достоинства автоматизации процессов проектирования:

1. Ускорение процесса проектирования: программы автоматизации способны быстро обрабатывать данные и строить модели, что позволяет значительно сократить время проектирования.

2. Снижение затрат: автоматизация проектирования позволяет сократить затраты на труд и материалы, уменьшить количество ошибок и лучше предсказать затраты на выполнение работ.

3. Увеличение точности и качества: автоматизация проектирования помогает избежать ошибок, связанных с человеческим фактором, и обеспечить более точные и качественные результаты.

4. Улучшение коммуникации: программы автоматизации способны обеспечить более эффективное взаимодействие между различными членами проектной команды, что улучшает коммуникацию и взаимодействие на всех этапах проекта.

Недостатки автоматизации процессов проектирования:

1. Необходимость обучения: программы автоматизации требуют обучения и опыта в использовании, что может занять время и вызвать неудобства для персонала, не имеющего опыта работы с такими инструментами.

2. Возможность ошибок: разработчики программ автоматизации могут допустить ошибки, которые могут привести к неверным результатам в процессе проектирования.

3. Необходимость постоянного обновления: программное обеспечение должно регулярно обновляться и поддерживаться для эффективной работы, что может привести к дополнительным затратам на обновление.

4. Ограниченность функций: программы автоматизации могут быть ограничены в функциях и возможностях в сравнении с ручным проектированием, особенно в специализированных областях.

В целом, автоматизация процессов проектирования является необходимой составляющей в современных условиях, где все больше уделяется внимание сокращению времени на проектирование и увеличению эффективности производства.

Автоматизация процессов проектирования началась еще в конце 1960-х годов с разработки программных продуктов, позволяющих автоматически генерировать чертежи, спецификации и технические отчеты на основе введенных пользователем параметров. В начале 1970-х годов появились первые системы компьютерного помощника (CAD), позволявшие создавать технические чертежи на персональных компьютерах. Одной из первых таких систем была AutoCAD, разработанная компанией Autodesk в 1982 году.

В 1990-х годах произошел значительный прорыв в области автоматизации процессов проектирования, связанный с развитием трехмерного моделирования (3D-моделирования) и систем управления жизненным циклом продукта (ПЛМ). 3D-моделирование позволило разработчикам создавать более точные и детальные модели продуктов, а системы ПЛМ – управлять процессами проектирования и производства продуктов на протяжении всего их жизненного цикла.

Сегодня автоматизация процессов проектирования является неотъемлемой частью любой инженерной отрасли. Существуют множество программных продуктов, позволяющих автоматически выполнить различные задачи при проектировании – от создания чертежей до анализа динамических и термических характеристик моделей. Компании, занимающиеся инженерными разработками, активно внедряют современные технологии автоматизации проектирования, чтобы повысить качество продуктов, ускорить процессы и снизить затраты на их производство.

Автоматизация процессов проектирования является одним из ключевых факторов повышения эффективности работы в различных областях промышленности, строительства и дизайна. Современные технологии и программное обеспечение, такие как CAD и CAE, позволяют значительно сократить время на проектирование и улучшить точность и качество проекта.

Кроме того, автоматизация процессов проектирования способствует уменьшению затрат на разработку, так как позволяет использовать стандартизированные и оптимизированные процессы и решения. Она также повышает гибкость и возможности внесения изменений в проект в любой момент, что является особенно важным при работе в условиях быстро изменяющихся требований рынка.

В целом, автоматизация процессов проектирования является незаменимой частью многих отраслей, которые стремятся повысить эффективность работы и снизить затраты на проектирование и разработку.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Справочник по САПР-системам: Технологии проектирования / И. В. Базылин. – М.: Радио и связь, 2005.

2. Автоматизация проектирования и технологии САПР / А.М. Головин, А.В. Корнеев, А.В. Рудаков. – М.: Новый Издательский Дом, 2016.

3. Современные методы и технологии САПР / Е.П. Катаев, Н.В. Горячева, А. В. Полежаев. – М.: Наука, 2013.

4. Современные технологии САПР: учебное пособие / В.Н. Иванов, В.А. Веригин, Р.А. Темперинский. – М.: Издательство МГТУ им. Баумана, 2014.

УДК 004.8

Е.А. БЛАГОДАРОВ
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ В ФОТОТЕХНИКЕ

Рассматриваются основные области применения искусственного интеллекта в создании фотографий и их последующей обработке.

В настоящее время искусственный интеллект (ИИ) является одной из наиболее обсуждаемых технологий в современном мире. Он применяется в различных сферах жизни, таких как медицина, учёба, транспорт, фототехника и т.д.

ИИ становится все более распространенным в фототехнике, помогая фотографам создавать более качественные и профессиональные снимки.

Использование искусственного интеллекта при настройке фотокамеры

Один из основных способов использования ИИ в фотографии – это автоматическая настройка экспозиции и баланса белого. ИИ может анализировать световые условия и настраивать камеру на оптимальные параметры, что позволяет сделать более яркие, контрастные и насыщенные фотографии.

1. Автоматическая настройка экспозиции.

ИИ может использоваться для автоматической настройки экспозиции фотокамеры в зависимости от условий освещения. ИИ может анализировать световые условия и настраивать камеру на оптимальные параметры, что позволяет сделать более яркие, контрастные и насыщенные фотографии.

2. Автоматическая настройка баланса белого.

ИИ может также использоваться для автоматической настройки баланса белого фотокамеры. ИИ может анализировать цветовую температуру освещения и автоматически настраивать камеру на оптимальные параметры, чтобы цвета на фотографиях выглядели естественно и точно отражали действительность.

3. Автоматическая настройка фокусировки.

ИИ может помочь автоматически настроить фокусировку на объекте съемки. ИИ может анализировать сцену и определять, на каком объекте нужно сфокусироваться, чтобы фотография выглядела более четкой и профессиональной.

4. Распознавание сцен.

ИИ может использоваться для распознавания сцен на фотографиях. ИИ может определить, какой тип сцены находится перед камерой (например, портрет, пейзаж, ночной город и т.д.) и автоматически настроить камеру на оптимальные параметры для создания наилучшей фотографии.

5. Распознавание лиц.

ИИ может также использоваться для распознавания лиц на фотографиях. ИИ может автоматически настраивать фокусировку на лице и применять эффекты, такие как смягчение кожи, улучшение контраста и яркости глаз, что делает портреты более привлекательными и профессиональными.

В целом, использование искусственного интеллекта для настройки фотокамеры позволяет фотографам создавать более качественные и профессиональные снимки, однако стоит учесть, что автоматическая настройка камеры не всегда позволяет добиться хорошего снимка, профессиональный фотограф может учитывать намного больше факторов, влияющих на результат.

Использование искусственного интеллекта при обработке изображений

Другой способ использования ИИ в фотографии – это автоматическая обработка изображений.

ИИ в обработке изображений – это технология, которая позволяет компьютеру анализировать и обрабатывать изображения, используя алгоритмы машинного обучения и нейронные сети. Это позволяет создавать программы, которые могут распознавать объекты на изображениях, классифицировать их, анализировать их содержание и даже создавать собственные изображения.

Одним из примеров использования искусственного интеллекта в обработке изображений является распознавание лиц. С помощью нейронных сетей и алгоритмов машинного обучения компьютер может анализировать изображения, определять на них лица и идентифицировать их. Это может быть полезно для систем безопасности, систем аутентификации и других приложений.

Применительно к обработке фотографий ИИ может улучшать качество снимков, удалять шум и размытие, улучшать резкость и цветовую гамму, что позволяет автоматизировать многие процессы

обработки упрощает жизнь фотографам и делает их работу более эффективной. Это особенно полезно в случаях, когда фотограф не имеет достаточного опыта в обработке изображений. Рассмотрим несколько способов применения ИИ при обработке фотографий.

1. Улучшение качества фотографий.

ИИ позволяет улучшить качество фотографий путем удаления шума и размытия изображения, улучшения резкости и насыщенности цветов. Это особенно полезно при работе с фотографиями, сделанными при недостаточном освещении или с использованием низкокачественной камеры.

2. Кроппинг.

ИИ может использоваться для автоматического выделения наиболее важной части фотографии и ее обрезки. Это позволяет сохранить наиболее важные детали и улучшить композицию изображения.

3. Реставрация старых фотографий.

ИИ может помочь восстановить старые фотографии, которые могут быть повреждены или иметь дефекты. ИИ может удалить пятна, царапины и другие дефекты изображения, восстановив его в исходное состояние.

4. Удаление объектов с фотографии.

ИИ может автоматически удалять объекты с фотографии, которые могут портить ее качество. Например, если на фотографии есть нежелательные элементы, такие как автомобили или люди, то ИИ может удалить их, сделав изображение более чистым и профессиональным.

5. Раскраска черно-белых фотографий.

ИИ может использоваться для раскраски черно-белых фотографий, делая их более реалистичными и привлекательными. ИИ может определить цвета, которые наиболее вероятно были использованы на фотографии, и автоматически раскрасить ее.

ИИ также может использоваться для улучшения портретных снимков. Он может распознавать лица на фотографиях и автоматически применять эффекты, такие как смягчение кожи, улучшение контраста и яркости глаз, что делает портреты более привлекательными и профессиональными.

ИИ также может быть использован для создания панорамных снимков. Он может автоматически склеивать несколько фотографий в одно большое изображение, что позволяет создавать эффектные панорамы без необходимости вручную настраивать камеру.

Наконец, ИИ может быть использован для распознавания объектов и сцен на фотографиях. Это позволяет автоматически размещать теги и описания на фотографиях, упрощая процесс каталогизации и поиска изображений.

В целом, использование ИИ в фототехнике позволяет значительно упростить настройку фотокамеры и обработку изображения, что ведет к ускорению процесса создания снимков. Будущее фотографии связано с использованием ИИ, и можно ожидать еще большего развития этой технологии в ближайшее время. Таким образом, использование искусственного интеллекта в фототехнике является актуальным и перспективным направлением развития.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Как искусственный интеллект применяется в фотографической среде. – Текст: электронный. – URL: https://pano-maker.ru/Articles/What-is-AI-Artificial-Intelligence-in-Photography/ (дата обращения: 15.04.2023).

2. Что такое AI (искусственный интеллект) в фотографии? Текст: электронный. – URL: https://blog.depositphotos.com/ru/iskusstvennyj-intellekt-i-budushhee-fotografii.html (дата обращения: 15.04.2023).

3. Пейзажи и портреты, созданные искусственным интеллектом Текст: электронный. – URL: https://photar.ru/pejzazhi-i-portrety-sozdannye-iskusstvennym-intellektom/ (дата обращения: 15.04.2023).

УДК 004.65

Е.А. БЛАГОДАРОВ
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ МАГАЗИНА ТЕХНИКИ

Рассматривается разработка информационной системы для магазина техники с помощью среды программирования MS Visual Studio (C#) и системы управления базами данных – MS SQL Server.

Интернет-магазины набирают все большую популярность у пользователей сети Интернет. С их помощью можно выбрать и заказать понравившийся товар, не выходя из дома. Как правило, интернет-магазины специализируются на определенных категориях

товаров, одной из которых является техника. Потенциальными клиентами магазина интернет-магазина офисной техники являются как частные лица, так и различные организации.

Создание программного обеспечения, реализующего функционирование интернет-магазинов, остается актуальной задачей.

В рамках статьи будет рассмотрена разработка клиент-серверного приложения для функционирования магазина техники.

Магазин техники – это предприятие розничной торговли, реализующее бытовую технику и электронику.

Средства разработки информационной системы:

- серверная часть – MS SQL Server;
- клиентская часть – MS Visual Studio (язык программирования C#).

Выявление функциональных потребностей пользователя

Приложение должно позволять потребителю с легкостью просматривать актуальные товары, добавлять их в корзину, оформлять заказ. Помимо этого в приложение должна быть реализована админ-часть. Сотрудник магазина может просматривать и обрабатывать текущие заказы, смотреть историю заказов, добавлять, удалять товары.

Полная use-case диаграмма представляет концептуальное представление бизнес-системы на этапе её проектирования. В ней описывается функционал будущей системы с указанием действий доступных конкретным группам пользователей.

В данной системе можно выделить следующие группы пользователей: Клиент, Администратор, Курьер.

Варианты использования системы группами пользователей различны.

Клиент может:

1. Просматривать и изменять личную информацию.
2. Просматривать каталог товаров.
3. Создавать или отменять заказ и т.д.

Администратор может:

1. Просматривать информацию о заказах.
2. Изменять статус заказа.
3. Просматривать и изменять каталог товаров и т.д.

Курьер может:

1. Просматривать информацию о закрепленных за ним заказах.
2. Просматривать информацию о клиентах, которые совершили его текущий заказ.

Полученная use-case диаграмма системы (рисунок 1):

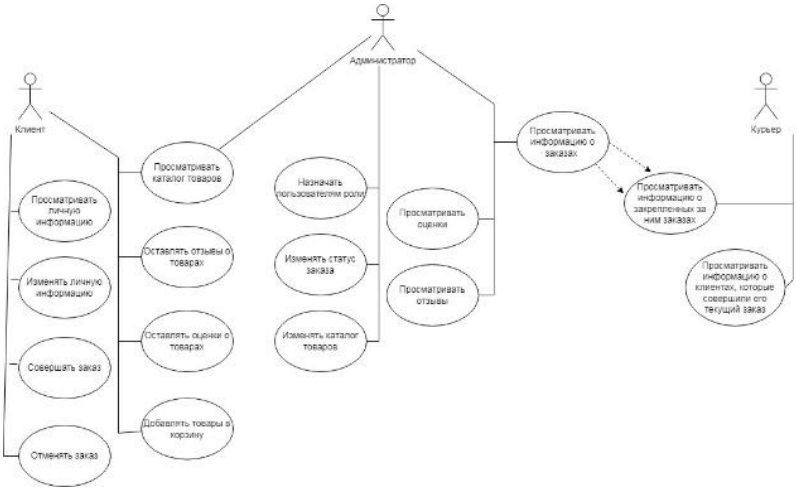


Рисунок 1 – Use-case диаграмма

Разработка общей структуры ИС

Архитектура информационной системы (ИС) – концепция, определяющая модель, структуру, выполняемые функции и взаимосвязь компонентов информационной системы.

Разрабатываемая ИС будет строиться по технологии клиент-сервер (архитектура представлена на рисунке 2). Она включает в себя клиентскую и серверную части и реализует следующие функции:

1. Управление данными в БД.
2. Обработка информации, которая хранится в БД.
3. Представление информации, которая хранится в БД.

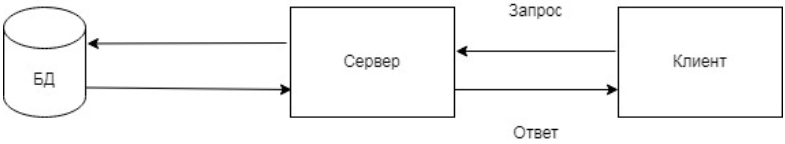


Рисунок 2 – Клиент-серверная архитектура

В данной модели пользователь отправляет конкретный запрос на сервер, где тот системно обрабатывается и результат отсылается клиенту. Сервер обрабатывает несколько клиентов в один момент времени. Если же одновременно поступает более одного запроса, то они устанавливаются в очередь и сервер выполняет их по очереди.

Разработка серверной части

В результате инфологического проектирования базы данных были выявлены сущности и связи между ними (с учетом существующих (бизнес-правил). Полная ER-диаграмма (рисунок 3).

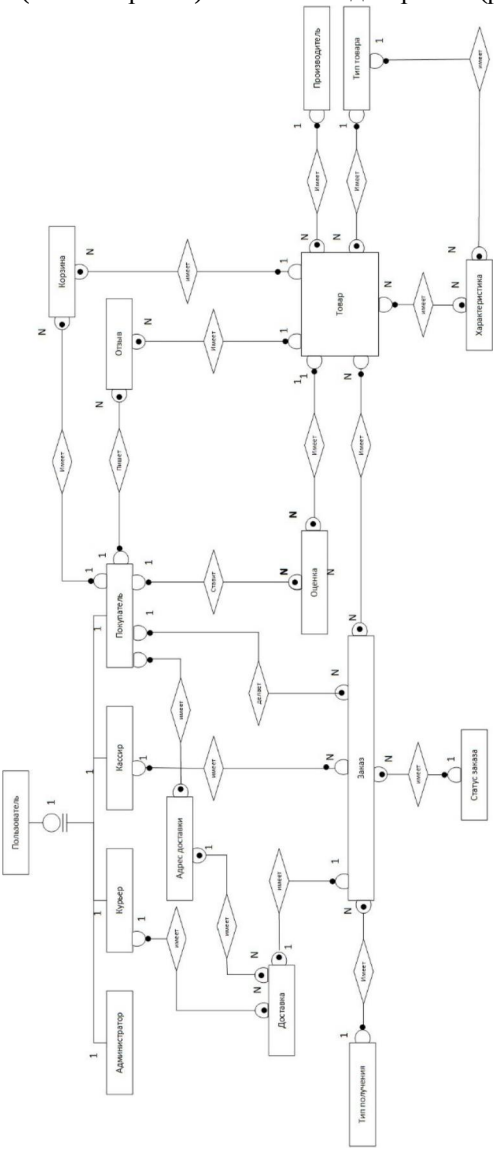


Рисунок 3 – ER-диаграмма

Следующим шагом осуществляется переход к предварительным отношениям в соответствии с правилами приведения к нормальным формам. Каждая сущность была проверена на соответствие нормальной форме Бойса-Кодда, а также построены диаграммы функциональных зависимостей для каждой сущности.

Для каждого атрибута каждой сущности был определен предварительный тип данных и требование к обязательности.

На основе проделанного проектирования была получена схема базы данных (рисунок 4).

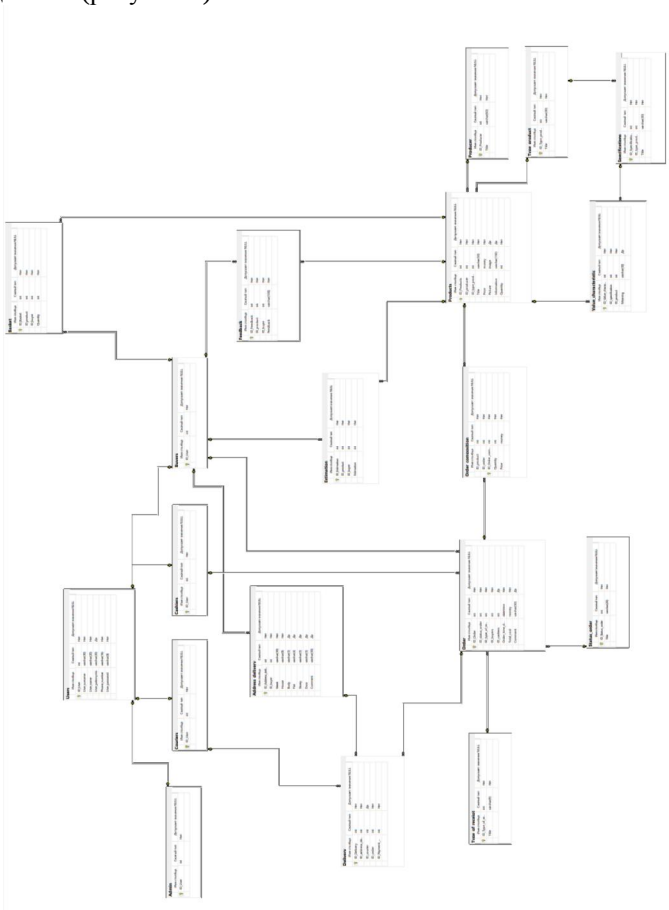


Рисунок 4 – Схема базы данных

Разработка клиентской части

На основе требований был разработан интерфейс приложения (в рамках статьи будет рассматриваться интерфейс администратора).

Начальное окно приложения (рисунок 5).

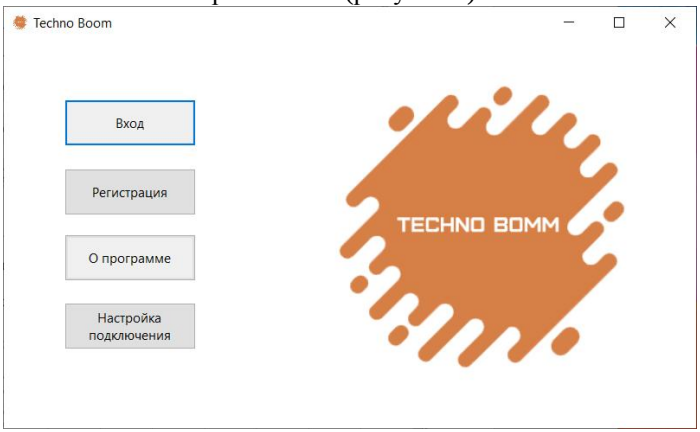
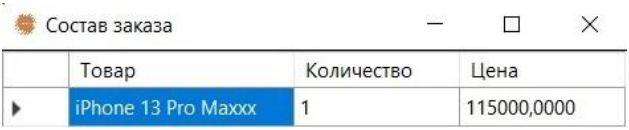


Рисунок 5 – Стартовое окно

На вкладке заказы отображены таблицы с информацией о заказе (рисунок 6). Двойной щелчок по заказу позволяет просматривать информацию о составе заказа (рисунок 7) и информацию о доставке.

Номер заказа	Фамилия	Имя	Отчество	Дата и время заказа	Телефон	Сумма заказа	Статус заказа	Тип доставки
1	Ольчев	Артем	Сергеевич	05.10.2022 12:36	89349812171	115000,0000	Обработка	Самовывоз
2	Ткачев	Даниил	Витальевич	05.10.2022 12:45	89133422290	112998,0000	Обработка	Самовывоз
3	Комарова	Маргарита	Алексеевна	05.10.2022 13:56	89782346766	113998,0000	Получен	Самовывоз
4	Благодарова	Тамара	Андреевна	05.10.2022 14:00	89324567782	386989,0000	Получен	Доставка
5	Миронова	Юлия	Сергеевна	05.11.2022 17:36	89005667890	69999,0000	Получен	Самовывоз
6	Карпухина	Дарья	Борисовна	05.11.2022 18:09	89236783455	263997,0000	Сборка	Самовывоз
7	Благодарова	Тамара	Андреевна	05.11.2022 18:45	89324567782	234999,0000	Готов	Доставка
8	Карпухина	Дарья	Борисовна	05.12.2022 9:15	89236783455	115000,0000	Готов	Доставка
9	Варфоломеев	Андрей	Максимович	05.12.2022 10:36	89001234567	230000,0000	Сборка	Самовывоз
10	Коротких	Илья	Сергеевич	05.12.2022 13:23	89567891268	167999,0000	Обработка	Доставка

Рисунок 6 – Вкладка заказов



Товар	Количество	Цена
iPhone 13 Pro Maxxx	1	115000,0000

Рисунок 7 – Окно просмотра информации о составе заказа

На вкладке товары располагаются информация о товарах (рисунок 8). Редактирование/добавление/удаление товара или характеристики происходит с помощью специальных кнопок (рисунки 8-9).

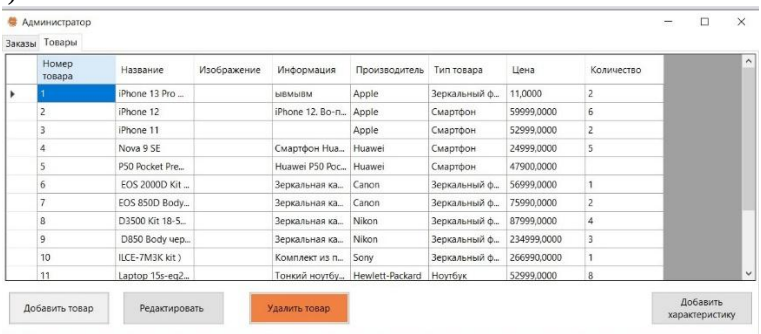


Рисунок 8 – Вкладка Товары

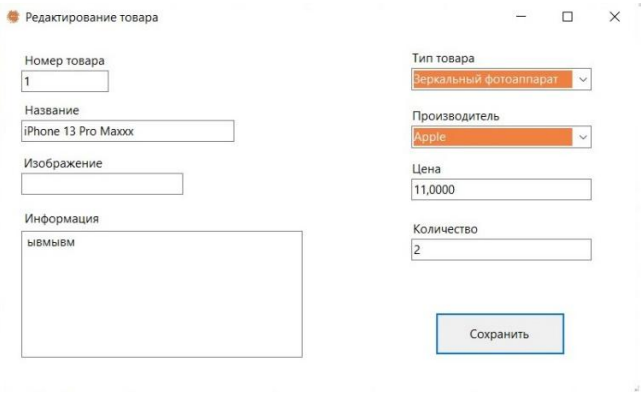


Рисунок 9 – Редактирование товара

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Нотация Питера Чена. – Текст: электронный. – URL: https://studme.org/77222/informatika/notatsiya_pitera_chena (дата обращения: 15.04.2023).

2. Гринченко Н.Н, Громов А.Ю., Благодаров А.В. Базы данных. Разработка клиентских приложений на платформе .NET. – 2018. – 287 с.

3. Албахари Джозеф, Албахари Бен C# 7.0. Справочник. Полное описание языка.: Пер. с англ. – ООО “Альфакнига”, 2018. – 1024 с.

УДК 004.056.55

А.А. ВАНЮКОВ
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ИССЛЕДОВАНИЕ МЕТОДОВ ОБЕСПЕЧЕНИЯ БЕЗОПАСНОСТИ ПРИ ПЕРЕДАЧЕ ИНФОРМАЦИИ

Рассматриваются методы обеспечения безопасности при передаче информации на примере мессенджера.

Использование мессенджеров на предприятиях

С появлением мессенджеров, связь между людьми стала более удобной и быстрой. Но не только для личной переписки эти приложения были созданы. Многие мессенджеры стали широко использоваться и в бизнесе. Они позволяют быстро и эффективно общаться с коллегами, делиться документами и информацией, а также координировать работу команды.

Разные мессенджеры для разных сегментов рынка

Существуют различные мессенджеры, которые нацелены на разные сегменты рынка. Telegram, например, является мессенджером для широкого круга лиц, в том числе и для бизнеса. Он позволяет создавать группы и каналы, в которых можно обмениваться информацией с коллегами и партнерами. Он также имеет функцию сохранения сообщений, что позволяет не потерять важные данные.

Slack является мессенджером, который разработан специально для бизнес-сегмента. Он позволяет создавать команды, каналы и чаты, где сотрудники могут обмениваться информацией и документами. Slack также имеет большое количество интеграций с другими приложениями, такими как Google Drive, Trello и другими, что делает работу с данными более удобной и эффективной.

SberChat — это корпоративный мессенджер компании Сбер. Он позволяет сотрудникам общаться между собой и с клиентами банка, а также предоставляет доступ к различным сервисам, которые могут быть полезными в работе банка.

Безопасность данных в мессенджерах

Однако, при использовании мессенджеров в бизнесе, необходимо учитывать безопасность данных. Мессенджеры могут использоваться не только для передачи информации между сотрудниками, но и для обмена конфиденциальной информацией с клиентами и партнерами. Поэтому, мессенджеры должны обеспечивать надежную защиту данных.

Как обеспечить безопасность данных в мессенджерах

Существует несколько способов обеспечить безопасность данных в мессенджерах.

- Шифрование сообщений. Многие мессенджеры используют шифрование для защиты конфиденциальной информации. Шифрование позволяет защитить сообщения от доступа третьих лиц, которые могут попытаться перехватить их в процессе передачи.
- Управление доступом. Мессенджеры могут предоставлять возможность управления доступом к чатам и группам. Так, только определенные сотрудники могут получить доступ к конкретным сообщениям или файлам.
- Аутентификация. Некоторые мессенджеры предоставляют возможность аутентификации пользователей, что позволяет убедиться в том, что сообщение было отправлено именно от того человека, от которого оно должно было прийти.
- Удаление сообщений. Некоторые мессенджеры позволяют удалять сообщения после отправки, что может быть полезно в случае, если сообщение было отправлено по ошибке или содержит конфиденциальную информацию.

Важно понимать, что безопасность данных должна быть приоритетом при использовании мессенджеров в бизнесе. При выборе мессенджера для использования в компании, необходимо убедиться, что он обеспечивает необходимый уровень безопасности.

Проанализируем методы шифрования, которыми может достигаться безопасная передача и хранение данных в мессенджерах.

Существуют различные методики шифрования данных:

- симметричное шифрование;
- ассиметричное шифрование;
- хеширование;
- сочетание методов шифрования.

Данные могут быть зашифрованы при помощи симметричного шифрования, что делает их нечитаемыми без соответствующего ключа. Ключ играет важную роль в этом процессе, так как он используется для шифрования и расшифрования данных. После шифрования данные могут быть безопасно переданы получателю, который может расшифровать их с помощью того же ключа. Поэтому, защита ключа является важнейшим аспектом симметричного шифрования, так как любой, кто имеет доступ к ключу, может расшифровать данные [1].

Безопасность ключа является наиболее уязвимым моментом в симметричном шифровании, как при его хранении, так и при передаче

между пользователями. В случае, если злоумышленник получит доступ к ключу, то он сможет легко расшифровать зашифрованные данные, полностью обесценив смысл шифрования. Еще одним недостатком симметричного шифрования является то, что программное обеспечение, которое обрабатывает данные, не может работать с зашифрованными данными. Для использования данных в таком программном обеспечении, они должны быть декодированы. Если же программное обеспечение скомпрометировано, то злоумышленник сможет легко получить доступ к данным. Однако такое шифрование при всей его уязвимости очень хорошо подходит для шифрования больших объемов данных ввиду несложности математических операций.

Асимметричный ключ шифрования отличается от симметричного ключа тем, что для кодирования и расшифровки сообщений используются разные ключи. Ключ для кодирования, или открытый ключ, известен всем пользователям сети, в то время как ключ для расшифровки, или закрытый ключ, остается тайным и используется только конкретным пользователем. Асимметричное шифрование также известно как шифрование с открытым ключом, и оно позволяет снизить вероятность расшифровки сообщения хакерами, поскольку секретный ключ не передается каждый раз. Примеры алгоритмов, использующих шифрование с открытым ключом, включают Diffie-Hellman и RSA. Этот тип шифрования очень плох для шифрования больших объемов данных из-за того, что математические операции, использующиеся в нем, требуют большого количества времени и являются более сложными для вычисления, чем операции, применяющиеся в симметричном шифровании.

Методика хеширования использует алгоритм, известный как хэш-функция для генерации специальной строки из приведенных данных, известных как хэш. Этот хэш имеет следующие свойства: одни и те же данные всегда производят тот же самый хэш, но изменение исходных данных хоть на один символ меняет получаемый хэш кардинальным образом. Попытки генерировать тот же самый хэш, изменяя входные данные, нецелесообразны. Основное отличие хэширования от других методов шифрования заключается в том, что после того, как данные захэшированы, они не могут быть восстановлены в исходном виде. Это гарантирует, что, даже если злоумышленник получит доступ к хэшу, он не сможет получить исходные данные. Два наиболее распространенных алгоритма хэширования – это Message Digest 5 (MD5) и Secure Hashing Algorithm (SHA).

Из-за ограничения в расшифровке данных хэширование обычно применяют для того, чтобы хранить те данные, которые никогда больше не понадобятся в первозданном виде (например, пароли).

Каждый из этих трех методов шифрования страдает от недостатков. Однако именно использование сочетания этих методов, образует надежную и эффективную систему шифрования.

Чаще всего, методики симметричного и ассиметричного шифрования комбинируются и используются вместе. Метод симметричного шифрования дает возможность быстрой шифровки и расшифровки, в то время как метод ассиметричного шифрования предлагает более безопасный и более удобный способ для передачи секретного ключа. Эта комбинация методов известна как «Сквозное шифрование» [2].

Мессенджеры являются удобным инструментом для общения и сотрудничества в бизнесе. Однако, при использовании мессенджеров необходимо учитывать безопасность данных. Различные мессенджеры предназначены для разных сегментов рынка, поэтому важно выбрать приложение, которое подходит именно для вашего бизнеса. При выборе мессенджера, необходимо убедиться, что он обеспечивает необходимый уровень безопасности данных и имеет необходимые инструменты для управления доступом к информации.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Швечкова О.Г., Пылькин А.Н., Марчев Д.В. Базовые криптографические алгоритмы защиты информации: учеб. пособие – М.: КУРС, 2018. – 168 с.

2. End-to-End Encryption, Secret Chats // Telegram: официальный сайт. – Текст: электронный. – URL: core.telegram.org/api/end-to-end (дата обращения: 11.04.2023).

УДК 004.65

Т.С. ВАСИЛЬЕВА, А.Н. САПРЫКИН
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ПРИМЕНЕНИЕ СОВРЕМЕННЫХ ИНТЕРФЕЙСОВ ВЗАИМОДЕЙСТВИЯ С БАЗОЙ ДАННЫХ В ВЕБ-ПРИЛОЖЕНИЯХ PHP

Рассматриваются современные интерфейсы взаимодействия с базой данных, а также их применение в веб-приложениях на языке PHP.

В настоящее время существует множество веб-интерфейсов для взаимодействия с базой данных (БД). Веб-интерфейс – это средство взаимодействия пользователя с веб-приложением через браузер. За счет обработки данных на стороне сервера с помощью языка PHP осуществляется упрощение доступа к БД. Встраиваемый в HTML-код и выполняющийся на сервере, данный скриптовый язык довольно хорош для разработки веб-ресурсов. Благодаря простоте кодирования и предоставлению высокой степени контроля над созданием качественного контента веб-программисту создание веб-приложений на PHP имеет ключевое значение.

Рассмотрим отдельно работу с БД через расширение PDO.

PDO

PDO является самым востребованным расширением для языка PHP. Благодаря особой безопасности расширения и наличию многофункционального интерфейса PDO высоко ценится среди разработчиков. За счет различных встраиваемых драйверов с помощью этого расширения возможно подключение практически к любой БД. Выделим следующие поддерживаемые им БД: MySQL, PostgreSQL, SQLite, Microsoft SQL Server, OCI, IBM DB2, ODBC и др.

Приведем строки для подключения к серверу MySQL:

```
$hbc1 = "mysql:host = $server_db; dbname =
$name_db; charset = utf8";
$connect_to_MySQL = new PDO($hbc1, $user,
$password);
```

Проиллюстрируем подключение к серверу ODBC с помощью строк:

```
$hbc2 = "odbc:MSSQLServer";
$connect_to_ODBS = new PDO($hbc2, $user,
$password);
```

Продемонстрируем строки для подключения к серверу OCI:

```
$hbc3 = "oci:dbname = $name_db; charset = utf8";
$connect_to_OCI = new PDO($hbc3, $user, $password);
```

Приведем подключение к серверу SQLite с помощью строк:

```
$hbc4 = 'sqlite:/[YOUR_PATH]/file.sqlite';
$connect_to_SQLite = new PDO($hbc4);
```

Иначе говоря, он позволяет осуществлять работу на различных серверах с помощью одинакового синтаксиса запросов. Если разработчик не определился под каким сервером будет работать его продукт, то стоит сделать выбор в пользу данного расширения. Он имеет собственный механизм соединения с БД, называемый источником данных.

У PDO есть свой механизм соединения с базой данных – «DSN» (Data Source Name), который принимает помимо основных данных тип

БД. Важным моментом является использование метода exec. В отличии от query, он возвращает количество записей, затронутых в SQL запросе. Представим подключение к БД MySQL с помощью PDO через фрагмент кода, который содержит обновление записей таблицы и вывод всех её строк. Приведем его ниже:

```
    $mech_dsn = "mysql:host=$server_db;dbname=$name_db;
charset=utf8";
    $mas = array(
        PDO::ATTR_ERRMODE  => PDO::ERRMODE_EXCEPTION,
        PDO::ATTR_DEFAULT_FETCH_MODE =>
PDO::FETCH_ASSOC);
    try {
        $conn_pdo = new PDO($mech_dsn, $user,
$password, $mas); }
    catch (PDOException $e) {
        die('При подключении возникла ошибка!: ' . $e-
>getMessage()); }
    $sql1 = 'SELECT * FROM `Magazine`';
    $res1 = $conn_pdo->query($sql1);
    while ($row = $res1->fetch()) {
        echo $row['n_mag'] . "<br>";
    }
    $sql2 = 'UPDATE `Magazine` SET `Price` = 2500 WHERE
`Id` = 9';
    $res2 = $conn_pdo->exec($sql2);
    $conn_pdo = NULL;
```

Рассмотрим работу с БД через другие расширения.

MySQLi

Интерфейс MySQLi является улучшенной версией устаревшего MySQL и частично совместим с ней. Можно отметить его высокую скорость работы, поддержку версиями PHP, безопасность на достойном уровне, а также популярность на хостинговых площадках.

Благодаря наличию объектно-ориентированного интерфейса работа с БД стала удобнее, ведь до его появления использовали интерфейс функций.

Опишем некоторый алгоритм для работы с БД при помощи функций интерфейса.

Открытие соединения с сервером БД осуществляется через метод mysqli_connect(). В качестве результата функция возвращает ресурс соединения. Данная функция имеет три аргумента: имя компьютера, имя пользователя и его пароль [1]. Покажем соединение с БД, приведя строку:

```
$connect_to_bd = mysqli_connect('server_db','user',
'password','name_db')or die("Подключиться к БД
невозможно!");
```

В вышеприведенной строке использована функция die(). Если нам не удастся подключиться через mysqli_connect() тогда функция die() закончит работу программы и выведет предупреждающее сообщение. [1]

Проверка успешности соединения с БД осуществляется с помощью метода mysqli_connect_error(). В качестве результата функция возвращает текст ошибки. На рисунке 2 представлен пример проверки соединения с БД.

```
if($connect_to_bd == false){
    print("Ошибка! Невозможно подключиться к БД" .
mysqli_connect_error());
}
else{
    print("Соединение успешно установлено!");
}
```

Крайне важно не забыть об **установке кодировки**, чтобы при обмене данными с БД не получить различные символы и знаки. Проиллюстрируем задание кодировки, приведя строку:

```
mysqli_set_charset($connect_to_bd, "utf8");
```

Формирование SQL **запроса** осуществляется через функцию mysqli_query(). В результате функция получит какие-то данные таблицы. Этой функции необходимо передать в качестве параметра строку с запросом. [1] Представим создание и выполнение SQL запросов с помощью следующего фрагмента кода:

```
$sql3 = 'SELECT* FROM `Kompl` WHERE `Firma` =
"HP"';
$res3 = $mysqli->query($sql3);
while($row = $res3->fetch_object()){
    print $row->n_komp .'<br>';
}
$sql4 = 'UPDATE `Kompl` SET `Price` = "1200" WHERE
`Number` = 6';
$res4 = $mysqli->query($sql4);
print($res4)?'Завершено!':'Ошибка (. $mysqli-
>errno .') '. $mysqli->error;
```

Закрытие соединения с сервером БД осуществляется через функцию mysqli_close(). Функция в качестве аргумента содержит переменную со строкой соединения с БД. Покажем закрытие соединения с БД с помощью строки, приведенной ниже:

```
$mysqli->close();
```

Подключение к БД средствами MySQLi показано через фрагмент кода, который содержит выборку данных из таблицы, а также вставку и обновление имеющихся записей. Приведем его ниже:

```
    $mysqli = new mysqli($server_db, $user, $password,
$name_db);//Попытка соединения с БД
    if (mysqli_connect_errno()) {
        printf("Сбой подключения: %s\n",
mysqli_connect_error());
        exit();
    }
    $mysqli->set_charset('utf8');
    $sql5 = 'SELECT* FROM `Theatre` WHERE `Actor` =
"Сидоренко Анна Гавриловна"';
    $res5 = $mysqli->query($sql5);
    while($row = $res5->fetch_object()){
        print $row->Name_actor .'<br>';
    }
    $sql6 = 'UPDATE `Theatre` SET `Salary` = "20000"
WHERE `Number_actor` = 9';
    $res6 = $mysqli->query($sql6);
    print($res6)?'Завершено!':'Ошибка (. $mysqli-
>errno .') '. $mysqli->error;
    $res5->free();
    $res6->free();
    $mysqli->close();
```

PgSQL

Несмотря на то, что интерфейс не имеет сильной известности и спроса среди веб-программистов, он обладает высокой надежностью и безопасностью в работе.

Опишем некоторый алгоритм для работы с БД при помощи функций интерфейса.

Открытие соединения с сервером БД выполняется с помощью функции pg_connect(). Она принимает строку соединения в качестве аргумента, возвращая дескриптор соединения с БД. [2] Проиллюстрируем соединение с БД, приведя следующую строку:

```
    $connect_to_bd = pg_connect("host=localhost
port=5452 dbname=Hospital user=postgres password=2002");
```

Проверка успешности соединения с БД осуществляется с помощью части кода, представленного ниже:

```
    if(!$connect_to_bd) {
        echo "Ошибка! Невозможно подключиться к БД\n";
    }
    else {
        echo "Успешное подключение к БД\n";
    }
```

Создание, а также выполнение SQL **запроса** осуществляется с помощью функции pg_query(). В качестве результата функция получает данные таблицы. Покажем создание и выполнение запроса, приведя следующий фрагмент кода:

```
$res=pg_query($connect_to_bd, "SELECT* FROM
Staff");
while($row = pg_fetch_object($res)) {
    echo "<br/>".$row->st_name." ".$row->st-salary;
}
```

В вышеприведенном фрагменте также присутствует функция pg_fetch_object(), позволяющая выбирать результирующие значения. Каждое имя поля будет представлено свойством данного объекта [2].

Закрытие соединения с сервером БД выполняется через функцию pg_close(). Она содержит переменную со строкой соединения с БД как аргумент. Функция не является обязательной, ведь PHP автоматически закрывает все открытые непостоянные соединения с БД. [2] Приведем строку, иллюстрирующую закрытие соединения:

```
pg_close($connect_to_bd);
```

Бесспорно, работа с БД достаточно заметная возможность языка PHP, ведь они заметно расширяют функционал веб-приложений. Благодаря наличию разных расширений для языка PHP веб-программист может выбрать более удобный вариант в зависимости от разрабатываемого продукта.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Можаров, М.С. Проектирование и разработка информационных систем с web-интерфейсом: учебное пособие / М.С. Можаров. – Новокузнецк: КГПИ КемГУ, 2019. – 135 с.

2. Стоунз Р., Мэтью Н. PostgreSQL. Основы. – Пер. с англ. – СПб: Символ-Плюс, 2002. – 640 с.

УДК 004.582

Д.А. ВЕНЧИКОВ, Д.С. ВЕНЧИКОВА
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ ПОМОЩИ ВОДИТЕЛЮ И ОПТИМИЗАЦИИ ТРАФИКА

Поскольку глобальный автопарк растет вместе с быстрорастущим населением планеты, разрабатываются новые решения для оптимизации дорожного движения.

Причиной для создания интеллектуальных систем управления дорожным движением стало недовольство общественности увеличением времени в пути при масштабной автомобилизации городских дорог. С развитием информационных технологий, которые упрощают нашу жизнь в различных мелочах, стало возможным применить их к делу, которое затрагивает всех нас ежедневно, – к дорожному трафику. Одной из инноваций стала технология ассистента водителя, которая призвана улучшить безопасность и комфортность вождения. Автоматический помощник позволяет водителю сосредоточиться на дороге, не отвлекаясь на монотонные задачи управления автомобилем. В этой статье рассмотрены возможностт ассистента водителя и его пользе для нашей жизни, но сначала немного истории.

Первая саморегулирующаяся транспортная система была применена перед Олимпийскими играми 1984 года в Лос-Анджелесе, сейчас она называется ATLAC (Автоматизированная система наблюдения и контроля за дорожным движением) и работает на 4600 перекрестках города. Согласно предоставленным данным, использование системы позволило на 16% увеличить скорость передвижения по городу и на 12% сократить время в пути. В 2009 году в Питтсбурге была запущена система интеллектуальных светофоров Traffic 21, разработанная в Университете Карнеги-Меллон, которая сократила время простоя транспортных средств на 40%, время в пути - на 26%, а выбросы вредных веществ – на 21%.

В Tyler, штат Техас, в 2010 году BMW и Siemens впервые запустили систему Intelligent Traffic System, которая отслеживает дорожную ситуацию и предлагает обходные маршруты водителям, у которых есть anti-idling технология. Несмотря на то, что небольшое количество автомобилей оснащено подобной технологией, это снизило задержки на дорогах на 22%. Если 30% участников дорожного движения получают информацию из умных источников, это значительно влияет на дорожную ситуацию в целом.

Система умного траффика представляет собой набор датчиков, расположенных на дороге, системы их связи, системы обработки поступающей информации и системы принятие решений регулирования траффика, а так же системы оповещения водителей.

Простейшие датчики, такие как индуктивные катушки, используются для определения наличия автомобилей над ними. Однако, они не могут отличить типы транспорта. Более сложные системы, такие как камеры, расширяют возможности датчиков и позволяют определять различные параметры движения на дороге. Для

сбора информации используются беспроводные протоколы, такие как LoRa, NB-IoT, Sigfox или RFID, которые передают данные на центральный командный центр для анализа и управления трафиком в реальном времени на определенных дорожных участках. Полученные данные помогают связать тенденции с управлением и оптимизацией потока движения. Подключение системы управления дорожным движением (светофоров и цифровых знаков) с их отображением на цифровой карте делает управление дорожным движением более четким и плавным.

Однако система, которая собирает о трафике с помощью датчиков, имеет существенные недостатки. Установка датчиков на дороге – дорогостоящий процесс. Датчики требуют обслуживания и настройки, что делает такую систему довольно затратной. Гораздо проще считывать информацию из смартфонов водителей, скачавших специальное приложение, в котором пользователь может заполнить необходимую информацию о своем автомобиле. Местоположение и скорость приложение сможет отслеживать по GPS. С помощью того же смартфона можно оповещать их ситуации на дороге, что гораздо надежнее и проще нежели осуществлять это посредством цифровых экранов на дороге. Мобильное приложение гораздо проще масштабировать, нежели сеть датчиков. В целом, приложение-ассистент для водителя может значительно облегчить жизнь водителя, улучшить безопасность на дороге и уменьшить эксплуатационные расходы. Ниже приведены некоторые функции, которые можно встроить в приложение-ассистент водителю:

1. Планирование маршрута с учетом текущей дорожной обстановки и наличия пробок.
2. Оповещение водителя о возможных аварийных ситуациях на дороге, которые влияют на трафик.
3. Автоматическое расчет расходов на топливо для заданного маршрута и типа автомобиля.
4. Информирование водителя о местах стоянки, бензоколонках и сервисных центрах на его пути.
5. Определение оптимальной скорости движения на основе факторов, таких как пробки, дорожные условия и экономия топлива.
6. Встроенный голосовой ассистент, который помогает водителю управлять приложением и получать информацию о местоположении и трафике.
7. Система уведомлений о близости к дорожным знакам и сигналам светофора.

8. Автоматический пересчет маршрута в случае изменения дорожной обстановки.

9. Интеграция с системой мультимедиа автомобиля для отображения информации на экране машины.

10. Автоматическое определение маршрута исходя из приоритетных задач водителя, таких как время прибытия, экономия топлива и наличие пробок.

11. Возможность управления приложением с помощью голосовых команд, чтобы водитель мог не отвлекаться от дороги и сохранять свою безопасность.

12. Интеграция с мобильными устройствами для отображения информации о трафике и текущей погоде на маршруте в режиме реального времени.

13. Возможность подключения к камере заднего вида автомобиля для повышения комфорта водителя при парковке.

14. Автоматическое уведомление об опасных участках дороги, таких как крутые повороты, острые рельефные перепады и штырьки на обочинах.

15. Система мониторинга состояния автомобиля, которая будет следить за работой двигателя, тормозных систем, электроники и других систем машины.

16. Интеграция с сервисом поиска и бронирования гостиниц на маршруте водителя, чтобы он мог быстро и легко забронировать жилье для ночевки.

17. Рекомендации по выбору маршрута на основе предпочтений водителя, таких как экономия времени, безопасность и пейзажи.

18. Автоматическая оплата дорожных сборов и парковок через приложение.

19. Возможность добавления заданий в список дел на маршруте, таких как покупка продуктов, поездка в банк и т.д.

20. Анализ статистики поездок для оптимизации маршрутов с учетом предпочтений и поведения водителя.

Этот список далеко не полный, применений данному приложению можно найти еще много. В итоге можно сделать вывод, что приложение-ассистент водителя является весьма перспективным направлением развития. Оно представляет собой не просто программу, а полноценного помощника, который сможет в значительной степени улучшить опыт вождения и повысить уровень безопасности на дорогах. Система автоматического анализа данных и распознавания обстановки на дороге, индивидуальный подход к каждому водителю и

широкий спектр функций приложения делают его уникальным и незаменимым инструментом для любого автолюбителя.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Огородников А.А. Автоматизация производственных процессов в автомобильной промышленности. – Москва: Транспорт, 2018.

2. Николаев А.А. Системы управления транспортными потоками. – Москва: Издательский дом «Транспорт», 2015.

3. Информационные технологии в автомобильной индустрии // Сборник научных статей, под редакцией Родионова Д.В. – Москва: Спутник+, 2019.

4. Автомобильная промышленность в эпоху цифровизации // Материалы международной конференции, под редакцией Иванова А.А. – Москва: Автопром, 2017.

5. Автоматизация процессов управления автотранспортом // Сборник научных статей, под редакцией Смирнова В.В. – Москва: Интерэксперт, 2016.

УДК 004.582

Д.А. ВЕНЧИКОВ, А.С. СТЕПЧЕНКОВ
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

РАЗРАБОТКА СЕРВИСА ДЛЯ СОВМЕСТНЫХ ПОЕЗДОК

Существенную часть своего времени люди проводят в пути. Но часто из-за пробок поездка превращается в неприятное времяпрепровождение. В этой статье рассматриваются некоторые аспекты разработки и применения сервиса для совместных поездок, призванный решить проблему с заторами.

Для повышения удобства перемещения жителей во многих городах Российской Федерации был введен общественный транспорт: автобусы, микроавтобусы, троллейбусы, метро и другие виды. Однако роль личного автотранспорта в городском трафике также весьма высока. В России на первое января 2022 года по данным ГИБДД 50,3 млн автомобилей и их количество растет на 1-2 процента в год из-за чего с каждым годом ситуация на дорогах РФ становится все сложнее. При этом также растут цены на бензин (литр бензина за последний год стал стоить в среднем на 35 % дороже). Рост зарплат граждан отстает от роста цен на топливо примерно в 3 раза (по данным Росстат с 2020

по 2021 год средняя зарплата по всем организациям в РФ увеличилась на 12,58%). Все этого говорит о том, что с каждым годом жителям нашей страны все сложнее обслуживать свой автомобиль. Все эти факторы способствуют тому, чтобы водители были более склонны к тому, чтобы экономить на поездках на личном транспорте. Те, кто пользуются общественным транспортом зачастую испытывают неудобства из-за его избыточной загруженности, и рисков инфицирования. Многие из этих людей предпочли бы поехать на такси, но не у всех есть достаточно средств для этого. Средний чек на поездку по сравнению с 2020 годом вырос с 300 до 350 рублей.

Если рассматривать абитуриентов, студентов и рабочих, которые проживают за пределами города, но не имеющих собственное авто, они вынуждены платить значительные суммы на поездки или подстраивать свои планы под маршрутное такси.

В связи с этим мы предлагаем использовать наш сервис, позволяющий собираться в группы и передвигаться по городу вместе. В этом случае водитель получает возможность заработать и сэкономить на топливе, а пассажиры получают комфортный транспорт для поездки в конечную точку, за оптимальную цену.

Программная информационная система под брендовым названием «Докинь» состоит из мобильного приложения, веб-приложения и вспомогательных сервисов. Рассмотрим веб-приложение более подробно. Архитектура приложения – это микросервисная архитектура. Основная сборка построена по принципу MVC, что позволяет эффективно вести разработку. Внутри контроллеров вызываются сервисы, что позволяет выделить дополнительный уровень абстракции при разработке и дополнительно отделить детали работы каждого сервиса от бизнес-логики.

Для написания веб-приложения и сервисов были использованы следующие технологии:

- Используемые языки программирования – C#, JAVA, JavaScript, HTML, CSS;
- Фреймворки – ASP.NET Core MVC.

Разворачивание приложения производилось посредством технологии контейнеризации docker. В качестве брокера сообщений используется Apache Kafka. Представленный выше стек технологий позволил сделать приложение устойчивым к увеличению трафика. Выбранная архитектура позволяет сделать удобной поддержку и дальнейшую модернизацию приложения.

Далее рассмотрим реализацию мобильного приложения. После исследования рынка было выяснено, что большая часть мобильных

устройств в России, используют операционную систему android. Поэтому было принято решение за основу взять ее, а реализацию кода написать на языке Kotlin.

После отключения платежной системы Swift оплата сервисов Google стала невозможной, и появился риск их блокировки в России, поэтому главной частью приложения является отечественная система Яндекс MapKit. Она обладает рядом преимуществ по сравнению с Google Maps:

- детальное покрытие России и стран СНГ;
- адреса более точные, лучше проработаны маршруты;
- одна цена на Геокодер, JavaScript API, Static API и MapKit SDK.

С ее помощью была выстроена интерактивная карта, алгоритм обнаружения пробок и поиска актуальных маршрутов.

Так как основная логика и вычисления производятся с помощью вспомогательных сервисов, то аппаратные требования к устройству, на который устанавливается приложение – минимальны.

Средой разработки стала Android Studio в качестве ее достоинств можно выделить:

- приятный дизайн пользовательских интерфейсов, позволяющий облегчить визуальное проектирование приложения;
- удобный XML редактор;
- эмуляция устройств;
- возможность проводить тестирование и анализ кода;
- скорость сборки приложения.

Основные преимущества сервиса для совместных поездок «Докинь»:

1. Широкий список настроек для пользователя, включающие логику формирования маршрута, выбор многочисленных критериев попутчиков.

2. Голосовой помощник водителя, позволяющий водителю проезжать маршрут максимально эффективно (алгоритмы сообщают о новых возможных попутчиках прямо по ходу движения).

3. Проработанный скоринговый сервис взаимодействующий с госуслугами, позволяющий сделать маршруты водителей и попутчиков максимально безопасными.

4. Интеграция с системой светофоров города, позволяющая давать рекомендации водителю, вплоть до корректировки скорости в зависимости от того, успевает он на следующий светофор или не стоит торопиться и сохранить ресурсы двигателя и тормозной системы (особенно актуально в ночные часы, когда трафик разгружен).

Далее описана часть основного клиентского пути пользователя сервиса.

При открытии приложения загружается интерактивная карта с возможностью вписать начальную и конечную точку маршрута. В результате поиска будут предложены уже запланированные маршруты водителями с указанием времени отправления, которые частично или полностью совпадают с желаемым. Система предложит войти в свой аккаунт или зарегистрироваться, чтобы записать пользователя в поездку. Так же возможна гибкая настройка попутчиков и водителей. Возможно указать количество попутчиков, пол водителя, отношение к курению, стаж, необходимость верификации водителя/попутчика по паспортным данным, вид автомобиля, необходимость детском кресле и многое другое. В личном кабинете есть возможность самому стать водителем, нажав на которую вы попадете на окно аутентификации, для заполнения данных о водительском удостоверении и управляемым им авто. После чего вам предложат создать свой собственный маршрут и указать количество человек, которые могут к нему присоединиться. Если маршрут совершается регулярно, можно поставить автосоздание заказа в указанное время и дни.

Таким образом, в ходе данной работы была выбрана и проработана архитектура сервиса для совместных поездок, определены технологии, используемые при разработке веб-приложения и мобильного приложения, а также был реализован минимальный клиентский путь, позволяющий найти попутчика или водителя с похожим маршрутом как в мобильном приложении, так и в веб-приложении. Также было произведена настройка базы данных и первичное разворачивание приложения. Были произведены настройки доступа из общедоступной сети и привязка доменного имени.

Были реализованы вспомогательные сервисы: сервисы регистрации и авторизации, сервис поиска совместных маршрутов, сервис настройки личного кабинета.

Далее планируется дальнейшая доработка сервисов и расширение функционала: интеграция с госуслугами и разработка скоринг-сервиса, настройка голосового помощника водителя и интеграция с сетью светофоров в городах РФ с целью оптимизации маршрутов. Также планируется разработка приложения под IOS.

Ниша сервисов для совместных поездок только начинает заполняться, поэтому в перспективе нескольких лет сервис «Докинь» имеет все шансы стать основой для успешного бизнеса.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Клеппман М. Высоконагруженные приложения. Программирование, масштабирование, поддержка. – СПб.: Питер, 2022. – 640 с.

2. Документация ASP .NET Core MVC. – Текст: электронный. – URL: https://metanit.com/sharp/aspnetmvc/1.1.php (дата обращения: 15.04.2023).

3. Документация Map Kit Sdk. – Текст: электронный. – URL: https://yandex.ru/dev/maps/mapkit/?from=mapsapi (дата обращения: 15.04.2023).

4. Микросервисы (Microservices). – Текст: электронный. – URL: https://habr.com/ru/post/249183/ (дата обращения: 15.04.2023).

УДК 004.032.26

Д.С. ВЕНЧИКОВА
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ВЫБОР НЕЙРОННОЙ СЕТИ ДЛЯ РАЗРАБОТКИ РЕКОМЕДАТЕЛЬНОЙ СИСТЕМЫ ВЫБОРА НАПРАВЛЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ АБИТУРИЕНТАМИ

В данной статье рассматривается выбор нейронных сетей в пользу рекуррентных нейронных сетей (RNN) для разработки рекомендательной системы выбора направления высшего образования абитуриентами.

Актуальной задачей высшего образования является привлечение выпускников школ и колледжей к поступлению в высшее учебное заведение, увеличение количества абитуриентов и прогнозирование их потока. Создание рекомендательной системы поможет улучшить и повысить качество приемной кампании и выявить сильные и слабые стороны в принятии управленческих решений относительно контрольных цифр приема на отдельные направления.

В настоящее время выбор будущей профессии, а, следовательно, и выбор направления для получения высшего образования, является одним из самых важных решений, которое принимают выпускники школ, а также выпускники средних специальных учебных учреждений, так как многие выпускники колледжей или техникумов, решившие получить высшее профессиональное образование, делают выбор в пользу иных специальностей, которым они обучились. Поэтому,

помимо помощи учреждениям высшего образования рекомендательная система также поможет абитуриентам в выборе наиболее подходящего направления, основываясь на интересах и цифрах вступительных экзаменов и аттестата.

Актуальность разработки такой рекомендательной системы для нашего университета заключается в том, что в настоящее время рынок труда имеет сильный перевес в сторону профессий, связанных с IT-технологиями. В связи с этим многие выпускники хотят связать свое обучение, а в будущем и профессию, именно с этим направлением, но при поступлении в технический ВУЗ вчерашнему школьнику достаточно сложно разобраться в нюансах каждого направления. Поэтому разработанная рекомендательная система поможет абитуриенту сделать правильный выбор.

Принцип работы рекомендательной системы заключается в том, что она собирает данные об абитуриентах и использует их для прогнозирования того, какие направления высшего образования наиболее подходят их личности и интересам. Для этого система использует алгоритмы машинного обучения, которые обучаются на основе данных о прошлых выборах абитуриентов и предоставляют рекомендации на основе этих данных.

Рекомендательная система может быть разработана как самостоятельное приложение или встроена в систему приемной комиссии высшего учебного заведения. В последнем случае система будет использовать данные о предыдущих студентах университета и анализировать их результаты, чтобы помочь будущим абитуриентам в выборе подходящего направления.

Глобальной целью работы является создание интеллектуальной рекомендательной системы, использующей алгоритмы и методы машинного обучения, с целью решения задач высшего образования. Существует множество типов нейронных сетей, которые могут быть использованы для обучения рекомендательной системы, такие как многослойные перцептроны, рекуррентные нейронные сети, скрытые модели Маркова, сверточные нейронные сети, глубокие нейронные сети. Однако, в процессе изучения вышеописанных видов нейронных сетей, было решено остановиться на рекуррентных нейронных сетях или RNN.

Рекуррентные нейронные сети (RNN) широко применяются в различных областях искусственного интеллекта, включая обработку естественного языка, распознавание речи и компьютерное зрение. В последнее время RNN также нашли применение в задачах рекомендательной системы. Они могут быть использованы для анализа

пользовательских интересов и предоставления персонализированных рекомендаций на основе этих интересов. Рекуррентные сети, в отличие от многослойных перцептронов, способны использовать свою внутреннюю память для обработки произвольно длинных последовательностей.

В этой статье мы рассмотрим, почему для разработки рекомендательной системы с помощью нейронных сетей подходят рекуррентные нейронные сети.

Рекуррентная нейронная сеть является классом искусственных нейронных сетей, которые обрабатывают последовательность данных, таких как текст, звук или временные ряды. В RNN каждый нейрон имеет вход, выход и скрытое состояние, которое заботится о сохранении контекста для последующего использования.

Рекуррентные нейронные сети, по сути, не сильно отличаются от обычных нейронных сетей и могут быть представлены в виде нескольких копий одной и той же сети, взаимодействующих друг с другом и передающих информацию на следующую итерацию (рисунок 1).

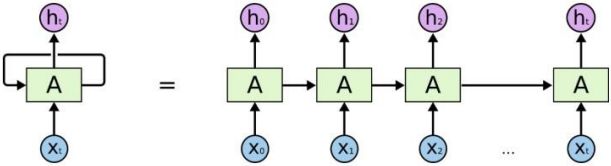


Рисунок 1 – Схематичное представление рекуррентных нейронных сетей

В RNN каждый нейрон получает входные данные и информацию из предыдущего скрытого состояния и использует эту информацию для вычисления нового скрытого состояния и выходных данных. Весь процесс повторяется для каждого элемента последовательности. В рекомендательных системах RNN могут использоваться для анализа и моделирования пользовательских интересов. Типичный подход к разработке рекомендательной системы с использованием RNN включает следующие этапы:

1. Представление данных.
2. Создание векторного представления материала: каждый элемент должен представляется вектором признаков.
3. Создание векторного представления пользователя.
4. Обучение модели: RNN может быть обучена предсказывать следующий элемент последовательности, основываясь на предыдущих элементах и векторном представлении пользователя.

5. Получение рекомендаций: на основе предсказаний модели можно получить рекомендации для пользователя.

RNN имеют несколько преимуществ, которые делают их привлекательными для использования в рекомендательных системах.

1. Универсальность: RNN могут обрабатывать данные произвольной длины и произвольного содержания, что делает их универсальным инструментом обработки последовательностей.

2. Интерпретируемость: скрытое состояние RNN можно интерпретировать как контекст текущего элемента последовательности, что может быть полезно для объяснения рекомендаций.

3. Адаптивность: RNN могут «запоминать» свои предыдущие состояния и использовать эту информацию для прогнозирования будущих состояний, что делает их адаптивными к изменениям в данных или вкусах пользователей.

Таким образом, в этой статье мы рассмотрели, почему для разработки рекомендательной системы с помощью нейронных сетей подходят рекуррентные нейронные сети. RNN могут быть использованы для анализа пользовательских интересов и предоставления персонализированных рекомендаций на основе этих интересов. RNN имеют несколько преимуществ, таких как универсальность, интерпретируемость и адаптивность, которые делают их привлекательными для использования в рекомендательных системах.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Копосов С.А., Копосова М.В. Обзор алгоритмов и задач рекуррентных нейронных сетей // Вестник Кемеровского государственного университета, 2018. – Вып. 2 (78). – С. 7-17.

2. Полосин А.В. Обучение рекуррентных нейронных сетей. Часть 1: Первоначальное знакомство с RNN. – Хабр, 2018.

3. Романина И.А., Мещеряков В.Я. Особенности обучения рекуррентных нейронных сетей для задач классификации и прогнозирования временных рядов // Информационные технологии и вычислительные системы, 2019. – Вып. 4 (174). – С. 38-51.

УДК 004.65

С.В. ГОЛОВИН
Рязанский государственный радиотехнический университет
имени В.Ф.Уткина

ОСНОВНЫЕ НАПРАВЛЕНИЯ РАЗРАБОТОК СИСТЕМ «SMART HOUSE» И КОНЦЕПЦИЯ «INTERNET OF THINGS»

Рассматриваются основные направления разработок систем «умный дом», включающие автоматизацию управления энергопотреблением, обеспечение безопасности и комфорта жильцов. Также рассматривают технические аспекты функционирования таких систем, их преимущества и возможности интеграции с другими устройствами IoT.

Развитие технологий не стоит на месте, и сегодня «Smart Home» больше не является фантастикой из кинофильмов. Технология IoT (Internet of Things) – это концепция взаимодействия объектов через Интернет, что позволяет автоматизировать различные процессы.

Термин «Internet of Things» обычно относится к тем сценариям, когда подключением к сети и вычислительными функциями оснащаются предметы, датчики и другие предметы повседневной жизни, обычно не считающиеся компьютерами, благодаря чему эти устройства могут генерировать и использовать данные и обмениваться ими при минимальном участии человека.

Основные направления разработок систем обеспечения «Smart House» можно классифицировать по следующим основным типам:

1) беспроводные и проводные;

2) централизованные и децентрализованные;

3) с открытым протоколом и с закрытым протоколом.

Каждый из представленных типов обладает достоинствам и недостатками. Рассмотрим примеры реализации проводных, беспроводных и децентрализованных типов технологий «Smart House».

Проводные системы были распространены до недавнего времени. Их особенность состоит в проводном соединении различных компонентов системы.

Примером является система «Smart House» с использованием протокола X10, предназначенная для интеллектуального управления освещением и бытовыми электроприборами при помощи силовой электропроводки. Сигналы управления X10 передаются и принимаются непосредственно по электрической сети. Для управления системой X10 производится передача команд управления с дистанционного пульта по радиоканалу с частотой 433МГц.

Достоинства проводных систем: высокая надежность и помехоустойчивость, большой срок службы и пожаробезопасность.

Недостатки проводных систем: установка такой системы возможна только на этапе строительства, низкое быстродействие системы, в один момент времени возможна передача только одной команды, при использовании в системе устройств с низким энергопотреблением (менее 50 Вт) сигналы от управляющего устройства могут искажаться или вовсе не доходить до адресата.

Беспроводные системы наиболее распространены на данный момент. В таких системах сигналы от управляющего устройства передаются к исполнительным устройствам по радиоканалам. В современных условиях это реализуется при помощи Wi-Fi-сети и подключения через сеть интернет.

Примерами беспроводной технологии служат такие известные системы, как:

- «Умный дом» от Яндекса;
- Amazon Echo;
- Google Home;
- Apple home kit.

Основой этих систем являются голосовые помощники «Алиса», «Алекса», «Google assistant» и «Siri» соответственно. При наличии программного и технического обеспечения они управляют бытовыми приборами при помощи Wi-Fi.

Достоинства вышеприведенных беспроводных систем: управление всеми устройствами при помощи голоса, легкое введение в эксплуатацию, большое количество подключаемых модулей.

Недостатки вышеприведенных беспроводных систем: отсутствие поддержки русского языка у Amazon Echo и Apple home kit, обработка всех данных происходит на серверах компании-разработчика, данные устройства работают с постоянно включенным микрофоном, безопасность.

Децентрализованные системы. Главная особенность – каждое устройство обладает собственным вычислительным устройством и энергонезависимой памятью.

Примером данной реализации являются системы, использующие коммуникационные шины KNX. Она была создана в результате работы более 15 европейских компаний. Основное преимущество – при выходе из строя одного из устройств система продолжает работать в штатном режиме без этого устройства. Этот протокол очень часто используется в странах Европы.

Достоинства вышеприведенной децентрализованной системы: надежность, популярность и поддержка.

Недостатки вышеприведенной децентрализованной системы: установка возможна только на этапе строительства, дороговизна комплектующих.

Пользовательский интерфейс Smart Home должен быть интуитивно понятным, легко управляемым и гибким. Организация дистанционного доступа обязательна, поскольку это позволяет контролировать работу устройств и систем даже во время отсутствия владельца. Также, рекомендуется использование технологий голосового и жестового управления, что позволит более естественным образом взаимодействовать с системой.

Кроме основных направлений разработок, существует еще один аспект, который становится все более значимым – это концепция IoT. Целью этой концепции является создание максимально интегрированных систем, которые бы работали как единый целый механизм. Она предполагает соединить все электронные устройства интернета вещей транслируют в Интернет, чтобы пользователи могли контролировать свои устройства в любое время, из любой точки мира. В рамках Smart Home, такая концепция применяется для автоматизации процессов.

Концепция объединения компьютеров, датчиков и сетей для мониторинга и управления устройствами существует уже несколько десятилетий. Тем не менее, недавнее слияние нескольких тенденций на рынке технологий приблизило «Интернет вещей» к повседневной реальности. В число этих тенденций входят:

- доступ из любой точки;
- широкое распространение сетей на основе протокола IP;
- экономические тенденции в области вычислительных систем;
- миниатюризация;
- достижения в области анализа данных;
- развитие облачных вычислений.

Для внедрения IoT используются различные технические модели обеспечения связи, каждая из которых имеет свои собственные характеристики. Четыре общих модели обеспечения связи, описанные комиссией по архитектуре Интернет:

- от устройства к устройству;
- от устройства к облаку;
- от устройства к шлюзу;
- совместное использование данных на сервере.

Эти модели демонстрируют гибкость возможностей подключения и использования устройств IoT.

Таким образом, сетевая топология системы IoT базируется на простых узлах, которые собирают и передают ограниченное количество информации к центральному контроллеру или шлюзу, который обеспечивает подключение к интернету и облачным сервисам. Узлы и шлюзы должны быть разработаны таким образом, чтобы свести к минимуму потребление энергии, обеспечить надежные сетевые соединения и расширить диапазон беспроводного подключения к сети, насколько это возможно. В основе системы IoT находится процессор или микроконтроллер, который обрабатывает данные и запускает программные стеки, которые подключаются к устройству беспроводной связи.

В заключение, можно отметить, что системы «Smart House» и концепция IoT – это предложения будущего, которые уже получают широкое распространение.

В заключение, можно отметить, что системы «Smart House» и концепция IoT – это предложения будущего, которые уже получают широкое распространение. Они дают возможность людям получить больше комфорта, безопасности и экономии, создавая интеллектуализированный дом. Однако, следует учесть потенциальные риски подключения к Интернету множества устройств и необходимость обеспечения безопасности персональных данных. Тем не менее, эти системы уже являются значимой частью современной жизни и в ближайшем будущем уверенно будут развиваться и улучшаться, предоставляя все более передовые и инновационные возможности для повседневного комфорта.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Макаров С.В., Гусев В.А. Smart House – технологии и возможности // Электротехнические системы и комплексы. – 2018. – №14(2). – С.50-59.

2. Иваненко В.А., Щербин Д.В. Автоматизированные системы управления домашней электротехникой // Электрика. – №3, 2018. – С.10-14.

УДК 004.65

С.В. ГОЛОВИН
Рязанский государственный радиотехнический университет
имени В.Ф.Уткина

КИБЕРБЕЗОПАСНОСТЬ: ОСНОВНЫЕ УГРОЗЫ И ЗАЩИТА ОТ НИХ

Предоставляет обзор наиболее распространенных угроз в киберпространстве, таких как вирусы, фишинг, хакерские атаки. Представлены методы, используемые злоумышленниками для атак на организации и предлагает меры, которые можно принять, чтобы защититься от них.

В современном мире технологий, кибербезопасность стала одной из наиболее значимых тем. С каждым годом количество киберугроз увеличивается, и они становятся все более усовершенствованными. В этой статье мы рассмотрим основные угрозы, которые могут влиять на кибербезопасность, а также расскажем о том, как их можно преодолеть.

1. Фишинг.

Фишинг – это мошенничество, в котором злоумышленник притворяется легитимной организацией, чтобы получить доступ к конфиденциальной информации, такой как логины и пароли. Обычно злоумышленники используют электронную почту, социальные сети и мессенджеры для отправки поддельных сообщений, в которых пользователей просят ввести свои данные в специальные формы. Фишинг-сайты могут быть крайне похожи на оригинальные, что делает их трудными для определения. Для защиты от фишинга необходимо следить за доменными именами сайтов, которые вы посещаете, а также проверять электронные письма, которые вы получаете, на предмет подозрительных ссылок и вложений.

2. Вредоносные программы.

Вредоносные программы – это программное обеспечение, которое использует компьютер для вредоносных целей. Вирусы, черви, трояны и шпионские программы – все это разные виды вредоносных программ. Они могут использоваться для кражи личных данных, уничтожения или шифрования данных, а также установки дополнительного вредоносного программного обеспечения. Вредоносные программы могут распространяться через электронную почту, социальные сети, мессенджеры, веб-сайты и т. д. Для защиты от вредоносных программ необходимо устанавливать антивирусное

ПО, которое регулярно обновляется, и не открывать подозрительные файлы и ссылки, которые могут содержать вредоносный код.

3. Социальная инженерия.

Социальная инженерия – это метод манипулирования людьми, чтобы они совершили некоторые действия. Это может быть, например, убедительная просьба передать конфиденциальную информацию, подделка электронных писем от руководства организации или угроза закрыть аккаунт. Для защиты от социальной инженерии необходимо обучать людей, как распознавать подобные атаки и быть осторожными в своих действиях.

4. Доступ к данным.

Доступ к данным – это угроза, при которой злоумышленники украли, взломали или получили доступ к персональным данным пользователей или организаций. Это может быть кража личных данных, финансовых данных, документов и другой конфиденциальной информации. Для защиты от угрозы доступа к данным необходимо использовать надежные пароли, создавать резервные копии данных и шифровать конфиденциальную информацию, передаваемую по сети.

5. Шифрование данных.

Шифрование данных – это процесс перевода обычного текста в зашифрованный вид, чтобы оригинальная информация была недоступна для злоумышленников. Шифрование может использоваться для защиты конфиденциальности личных данных, банковских данных и компьютерных файлов. В настоящее время существует множество инструментов для шифрования данных, таких как VPN-сервисы и приложения для управления паролями.

6. Недостаточная безопасность сети.

Недостаточная безопасность сети – это нарушение безопасности, которое может привести к перехвату конфиденциальной информации, а также к взлому устройства. Недостаточная безопасность сети может быть вызвана не только ненадежностью оборудования, но и несоблюдением правил безопасности при использовании сети. Для повышения безопасности сети нужно использовать сильные пароли и протоколы шифрования, а также обновлять ПО и настраивать защиту от внешних атак. Хакеры используют различные методы атак на сеть, такие как подделка MAC-адресов, атаки на SSL-шифрование и использование уязвимостей в ПО. Использование безопасных сетевых настроек и расширений браузера может предотвратить большинство таких атак.

7. Майнинг и DDoS-атаки.

Майнинг – это процесс, когда злоумышленник использует ресурсы компьютера для майнинга криптовалюты. Незаметно для пользователя, майнер отбирает ресурсы устройства и нагружает его процессор, что может привести к серьезным проблемам с работой устройства. Для защиты от майнинга нужно использовать сложные пароли и двухфакторную аутентификацию для защиты контрольной панели хостинга.

DDoS (от английского Distributed Denial of Service) – это атака на сервер, когда злоумышленник посылает на сервер такое количество запросов, что он перегружается и перестает отвечать на нормальные запросы. Это может привести к отказу в обслуживании и потере доступа к ресурсам.

Защита от угроз в области кибербезопасности

Существует несколько методов защиты от угроз в области кибербезопасности. Один из наиболее эффективных методов – это повышение осведомленности о существующих угрозах и мерах защиты. Пользователи и сотрудники организаций должны знать основы кибербезопасности, такие как использование надежных паролей, проверка адресов электронной почты и ссылок, создание резервных копий данных и т.д.

Также важно использовать различное программное обеспечение, которое защитит данные от угроз. Это может быть антивирусное программное обеспечение, фаерволы, программы для шифрования данных и т.д. Некоторые организации могут использовать специальные системы безопасности, такие как системы обнаружения вторжения или системы защиты от DDoS атак.

Кроме того, для защиты от угроз необходимо регулярно обновлять программное обеспечение и операционные системы, чтобы установить все доступные обновления и исправления уязвимостей. Это поможет предотвратить несанкционированный доступ к данным.

Наконец, важно проводить регулярные проверки систем на наличие угроз и осуществлять мониторинг защищенной информации. Это позволит быстро обнаружить подозрительную активность и предотвратить потенциальную угрозу.

Основные достоинства современной кибербезопасности:

1. Защита от киберпреступников и киберугроз – современные технологии позволяют быстро и эффективно определять уязвимости в системе и настраивать защиту от атак.

2. Возможность мониторинга сетевой активности – с помощью специального программного обеспечения можно отслеживать общий трафик и предупреждать о любых подозрительных активностях.

3. Автоматизация обнаружения и реакции на угрозы – на основе анализа больших данных можно создавать автоматизированные системы для обнаружения и быстрого реагирования на киберугрозы.

4. Быстрая защита – кибербезопасные системы позволяют быстро и эффективно обнаруживать и устранять угрозы, что помогает сократить время реакции и минимизировать убытки.

Основные недостатки современной кибербезопасности:

1. Сложность реализации и настройки – кибербезопасность требует скрупулезного подхода к настройке систем и проведению регулярных проверок и аудитов.

2. Затратность – создание и поддержание современных систем кибербезопасности требует значительной финансовой и человеческой энергии.

3. Возможность ошибок – любая система кибербезопасности подвержена ошибкам и уязвимостям, которые могут стать дверью для киберпреступника.

4. Недостаточное обучение персонала – многие угрозы связаны с недостаточной компетенцией персонала, работающего с кибербезопасными системами. Необходима регулярная образовательная работа для повышения уровня знаний и навыков.

Таким образом, кибербезопасность является одним из важнейших вопросов в нашей жизни. Но с правильными знаниями и инструментами, у нас есть возможность защитить свои устройства, данные и личную информацию. Важно не только знать основные угрозы, но и следовать основным правилам безопасности в интернете.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Степанов, А.В. Кибербезопасность: теория и практика защиты информации / А.В. Степанов. – М.: ДМК Пресс, 2016. – 416 с.

2. Федеральное агентство по техническому регулированию и метрологии (Росстандарт). Руководство по управлению кибербезопасностью информационных технологий. – Москва, 2020. – 40 с.

3. Перков, С.Е. Кибербезопасность: основы и принципы защиты / С.Е. Перков. – М.: Инфра-М, 2016. – 240 с.

УДК 004.42

М.С. ДУБЧЕНКО
Рязанский государственный Радиотехнический университет
имени В.Ф.Уткина

МОДЕЛИРОВАНИЕ ТРЕХМЕРНЫХ ОБЪЕКТОВ И СЦЕН В UNITY3D

Рассматриваются основные понятия движка Unity3D.

Unity3D – один из самых гибких и удобных движков для разработки игр, которые могут быть запущены на разных платформах, таких как Android, iOS, Windows и многих других. Вдобавок к этому, Unity3D носит кроссплатформенный характер, что значительно облегчает работу разработчикам в процессе создания мультисредних игр.

Данный движок также имеет обширную базу знаний, сообщества и технической поддержки, так что разработчикам доступны всевозможные материалы и руководства в онлайн-режиме, чтобы помочь им в ускорении и облегчении их работы.

Одним из главных преимуществ Unity3D является его силовой механизм, позволяющий разработчикам создавать эффекты физических взаимодействий, таких как столкновение и гравитация. Благодаря использованию принципов механики, игры на движке Unity3D становятся более реалистичными, а также являются более привлекательными для пользователей.

Другой важной особенностью Unity3D является возможность быстрого и простого создания 2D и 3D-анимации. Разработчики могут легко создавать новые персонажи, объекты и трансформировать их в соответствии своими потребностями. Оказывается, что 2D-графика размещенная в 3D-месте выглядит более единообразной и реалистичной, чего не удается достичь в других движках.

Unity3D также снабжен поддержкой мультиязычности, что делает его более гибким и адаптивным для глобальных рынков, где необходимо создавать приложения, доступные на разных языках. Приложения, созданные на Unity3D, могут быть запущены на любом языке, что будет полезным для тех, кто ищет международный успех в разработке приложений.

Кроме этого, Unity3D также имеет большой спектр пакетов расширения, которые значительно упрощают работу разработчикам на разных платформах. Например, для разработчиков мобильных игр на Unity3D существует множество расширений, которые помогают ускорить разработку, а также создавать ячейки для социальной сети, аналитические инструменты и многое другое.

Альтернативы Unity

Главный конкурент Unity среди бесплатных движков – это Unreal Engine, хотя есть еще и другие альтернативы – Godot Engine или Roblox.

Есть и закрытые движки, которые разрабатывают и поддерживают большие студии. Используя свой собственный движок, они не платят никакие отчисления другим компаниям и полностью контролируют его функциональность.

Основное различие Unity и Unreal Engine – это язык программирования: в Unity это C#, в Unreal Engine это C++. Кроме того, Unreal Engine более требователен к характеристикам компьютера.

Игровые движки позволяют рендерить в режиме реального времени и видеть результат работы сразу, на лету. Именно по этой причине они пользуются популярностью во многих областях, связанных с компьютерной графикой. Unity используется для создания игр, в архитектуре, автоиндустрии, кинематографе.

К преимуществам Unity3D можно отнести:

1. Функциональность графического редактора. У него присутствует возможность создания локаций, модулей, расстановку элементов на сцене с тестингом.
2. Кроссплатформенность. Игра может быть адаптирована не только для ПК, но и для консолей.
3. Интегрированная среда разработки Unity позволяет делать настоящие шедевры с минимальными навыками в сфере коддинга.
4. Поддержка плагинов.
5. Модульность. За счет нее можно конструировать пакеты компонентов в пределах одной игровой сценки.
6. Тщательно продуманная методика создания элементов.
7. Использование в основе создания игры C#. Он быстро осваивается и легко читается.
8. Поддержка Unity. На сайте и тематических ресурсах полно документации по Unity3D. Она поможет лучше изучить движок.
9. Бесплатное распространение. За счет этого любителям и новеньким не придется вовсе тратиться на контент.

10. Конечный результат указывает на то, что Юнити – более совершенная среда создания программного обеспечения и развлекательного софта.

Но недостатки у движка тоже есть. Среди них:

1. Сложности при написании многокомпонентных элементов.
2. Трудно подключать внешние плагины.
3. Не сразу получается корректировать шаблоны экземпляров.
4. Не лучшее быстродействие при использовании WebGL-редактора.
5. Большой размер итогового проекта.
6. Для того чтобы воспользоваться полным инструментарием Unity, требуется приобрести платную версию. Она стоит от 1800 долларов США. Enterprise-релиз обойдется вовсе в 4000 долларов.

Unity3D является одним из самых распространенных и мощных движков для создания игр и приложений в трехмерном пространстве. Для создания 3D-моделей и сцен в Unity доступны различные инструменты. Мы вспомним, как создавать объекты, работать с камерой, настраивать источники света, настраивать материалы и многое другое.

1. Создание объектов.

Наиболее очевидный способ создания объектов в Unity – использовать готовые примитивы, такие как куб, сфера, цилиндр и т.д. Но они могут быть грубыми и ограниченными. Поэтому, чтобы создавать более сложные объекты, можно использовать моделирование 3D-объектов во внешнем приложении, таком как Blender, ZBrush, 3D Max и Maya и экспортировать их в формате OBJ или FBX.

С другой стороны, Unity имеет встроенные инструменты для создания более сложных объектов. Например, проводя манипулятором по кривой, можно создать объект в форме требуемого контура. Каждая точка манипулятора становится вершиной объекта. Инструменты имеют также панель настроек, где можно настроить параметры объекта, такие как радиус, высота, количество сегментов, а также дополнительные параметры, в зависимости от типа объекта.

Создание собственных объектов позволяет точно передавать требуемый дизайн.

2. Работа с камерой.

Камера – это основной способ управления взглядом игрока в Unity. Она определяет, что и как показывать на экране. Камеры используются для отображения сцены на экране, а также для создания различных эффектов, таких как приближение, удержание и движение.

В Unity можно перейти к редактированию камеры, щелкнув на ней в иерархии объектов. Здесь мы можем настроить настройки камеры, такие как угол обзора камеры, размер экрана, дальность отрисовки и многое другое.

3. Работа со светом.

В Unity свет является важным составляющим в создании 3D-сцен. Сверху в панели инспектора можно выбрать тип освещения, например, realtime, чтобы управлять природным светом в игре. Дирекционное освещение используется для передачи солнечного света со своеобразным теневым эффектом. Точечные источники освещения используются для создания точечных ламп, таких как фонари и светильники.

Для каждого типа света доступны параметры, такие как яркость, цвет и расширение. Важно настроить свет таким образом, чтобы он максимально приближался к естественному свету. Здесь можно экспериментировать с разными настройками.

4. Настройка материалов.

В Unity3D материалы определяют внешний вид объектов: цвет, оттенок, текстуры, прозрачность и многое другое. Для создания материала можно выбрать на панели проекта, пкм по пустому месту и выбрать "Create/Material". Далее можно настроить параметры материала, такие как цвет, отблеск, глянец, масштаб и т.д. Использование текстур в материалах может существенно улучшить визуальное восприятие объекта.

Важным моментом при создании материалов является установка правильного типа тени. В Unity доступны типы теней, такие как hard, soft или transparent. Каждый тип тени подразумевает разные настройки и может быть использован для создания различных эффектов.

5. Оптимизация производительности.

Высококачественное моделирование трехмерных объектов и сцен в Unity3D – это комплексный процесс, который может потребовать много ресурсов от компьютера. Поэтому важно оптимизировать проект для более быстрой работы и лучшей производительности.

Одним из ключевых механизмов оптимизации в Unity является сокращение количества отображаемых элементов на экране. Для этого можно использовать технику отсечения объектов или LOD, когда менее важные объекты вдали отображаются в урезанной форме. Это позволяет снизить количество отображаемых полигонов и нагрузку на процессор.

Нужно также уменьшать количество текстур и материалов, которые применяются к объектам на сцене. Некоторые текстуры могут приводить к значительному увеличению размера проекта, поэтому необходимо использовать оптимальные форматы.

Кроме того, важно настроить камеру и источники света таким образом, чтобы сократить нагрузку на процессор. Использование объединения объектов и правильной настройки типа теней также могут помочь в оптимизации.

В заключение, моделирование трехмерных объектов и сцен в Unity3D – это комплексный процесс, который включает в себя работу со светом, тенями, камерой и материалами. Не менее важно оптимизировать проект, чтобы он работал стабильно и быстро. Мы рассмотрели основные инструменты и техники, которые помогут вам создать высококачественный проект в Unity3D.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Руководство пользователя Unity (2018.3). – Текст: электронный. – URL: https://docs.unity3d.com/ru/2018.4/Manual/index.htmlhttps://docs.unity3d.com/ru/2018.4/Manual/index.html (дата обращения: 15.04.2023).

2. Руководство пользователя Unity 2021.3 (LTS). – Текст: электронный. – URL: https://docs.unity3d.com/ru/2018.4/Manual/index.html (дата обращения: 15.04.2023).

3. Ходос О.С., Баженов Р.И. Обучение трехмерному моделированию в Unity3D // Современные научные исследования и инновации. – 2014.

УДК 004.338

А.В. ЕЛИСЕЕВ
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ИСПОЛЬЗОВАНИЕ ТЕХНОЛОГИИ BLOCKCHAIN

В статье рассматривается понятие и структура технологии blockchain, а также области её применения.

С каждым годом сфера информационных технологий модернизируется сильнее. Одна из лидирующих позиций в области развития IT является схема blockchain.

Blockchain – непрерывная последовательная цепочка блоков выстроенная по определённым правилам. Общая схема блоков и связей между ними представлены на рисунке 1.

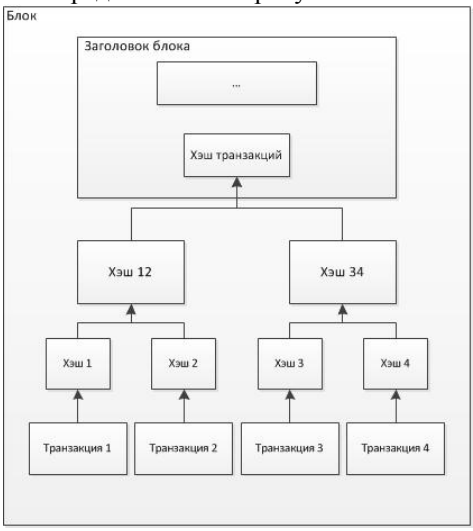


Рисунок 1 – Схема получения хэша транзакций

Блоки содержат необходимую информацию. Между ними существует связь в виде нумерации, собственной хэш-суммы блока и хэш-суммы предыдщуего блока. При изменении информации меняется и хэш-сумма в блоке. Для того, чтобы правильно выстроить цепочку необходимо соблюдать определённый алгоритм. Изменения хэш-суммы следует записать в следующий блок. Вследствие этого меняется его собственная хэш-сумма, а предыдущие блоки не будут затрагиваться. В случае изменения последнего блока в цепочке, внесение изменений не будет слишком затратным. В том случае, если после изменяемого блока создано продолжение, трансформация может быть достаточно трудоёмкой. Стоит добавить то, что копии цепочек блоков хранятся на разных компьютерах независимо друг от друга [1].

История возникновения рассматриваемой технологии берёт своё начало в 1991 году. В это время учёные исследователи Стюарт Хабер и У.Скотт Шторнетта внедрили решение вычислительно-практического характера для документов цифровой формы с штампом времени, в целях невозможности оформления их задним числом или подделки.

Идея системы заключалась в криптографической закреплённой цепочке блоков, где использовалось хранение документов с временной

отметкой. Данная технология не получила высокого спроса и не была запатентована.

После этого в 2004 году учёный в области компьютерных технологий Хэл Финни предоставил систему RPoW (Reusable Proof Of Work). Новая технология работала, получив незаменяемый или невзаимозаменяемый Hashcash токен, основанный на proof of work и подписанный в RSA, который впоследствии мог быть передан от человека к человеку.

В тот момент стояла проблема двойного расходования, то есть повторная продажа одних и тех же активов. Использование системы RPoW позволило решить этот вопрос. В итоге право собственности на токены, зарегистрированные на доверенном сервере, сохранялись. Данный сервер был создан для того, чтобы пользователи по всему миру смогли удостовериться в его правильности и целостности в режиме реального времени.

Впоследствии система RPoW послужила ранним прототипом в истории создания криптовалюты [2].

Развитие blockchain играет важную роль в снижении рисков безопасности, искоренения операций мошенников, а также в обеспечении прозрачности масштабируемым образом.

Выраженная популярность технологии blockchain пришла в момент появления криптовалюты и NFT. Пионером в сфере рынка криптовалют является bitcoin, созданный Сатоси Якомото и представленный в 2009 году.

Остановимся немного подробнее на трёх ключевых моментах технологии blockchain. Как и говорилось ранее, система имеет блоки, узлы, а также в эту структуру добавляются и майнеры.

Цепочка состоит из блоков, а блок состоит из трёх основных элементов:

1. Различные данные в блоке.

2. Nonce – «номер, использующийся только один раз». Одноразовый номер в blockchain представляет собой целое число, которое генерируется случайным образом, после чего появляется в хэше заголовка блока.

3. Хэш – хэш в blockchain – число, постоянно прикреплённое к одноразовому номеру. При создании первого блока цепочки, одноразовый номер создаёт криптографический кэш. Данные в блоке считаются подписанными и навсегда привязанными к одноразовому номеру или хэшу.

Пример криптографической хэш-функции приведён на рисунке 2.

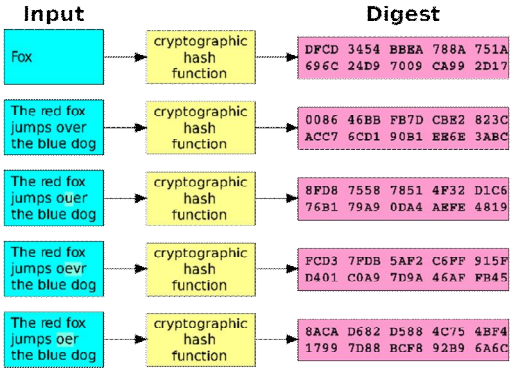


Рисунок 2 – Криптографическая хэш-функция

Майнер, работающий в сфере криптовалюты, создаёт новые блоки в цепочке с помощью майнинга.

Как и упоминалось ранее, блоки в цепочке имеют свой одноразовый номер и хэш, но они могут ссылаться на хэш предыдущего блока в цепочке, поэтому добыча блока является непростой задачей. Для решения очень сложной математической задачи поиска одноразового номера, генерирующего принятый кэш, майнеры используют специальное ПО. Одноразовый номер состоит из 32 бит, а хэш из 256, поэтому существует около четырёх миллиарда возможных комбинаций одноразового кода и хэша, которые необходимо обработать, прежде чем будет найдена правильная. При успешно добытом блоке происходит принятие изменений всеми узлами в сети, после чего майнер получает финансовое вознаграждение.

Следующей особенностью blockchain является децентрализация. Смысл этого термина в том, что компьютер или организация не могут владеть сетью. Вместо этого используется распределённый реестр через узлы, подключённые к цепочке.

Отличие централизованной от децентрализованной сети представлено на рисунке 3.

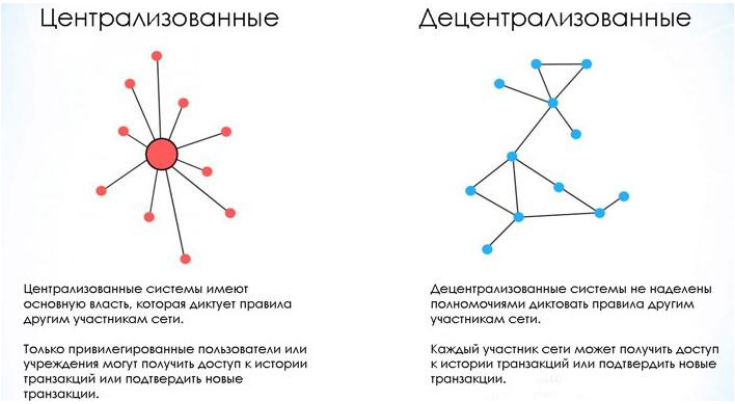


Рисунок 3 – Отличия централизованной и децентрализованной сети

Узлами blockchain могут быть любые электронные устройства, хранящие копии цепочки и обеспечивающие функционирование сети.

Отдельно стоит добавить про blockchain-платформу. Сеть blockchain описывает инфраструктуру распределённого реестра, платформа blockchain описывает среду, в которой пользователи могут взаимодействовать с blockchain и его сетью.

Достаточно популярным примером платформы blockchain является программная платформа Ethereum, на которой размещена одноименная криптовалюта. Данная платформа позволяет пользователям создавать программируемые токены и смарт-контракты, построенные на инфраструктуре непосредственно blockchain Ethereum [3].

Помимо использования блокчейна в криптоиндустрии, технология смогла себя зарекомендовать и в других областях. К примеру, в сфере смарт-контрактов. Это цифровые контракты, информация о которых сильно зашифрована. Главное отличие смарт-контрактов является автоматический контроль и выполнение пунктов договора. Если условия выполняются, контракт завершается автоматически, без совершения дополнительных действий и участия юристов. Преимущество данных контрактов в том, что они позволяют отслеживать всю цепочку поставок, что снижает или полностью исключает возможность подделки продукции.

Технология получила признание и в области NFT. Это вид токенов, где каждый экземпляр уникален, его нельзя заместить или обменять на другой токен. NFT свидетельствует о праве владения каким-либо активом в блокчейне и в последствии появляется

возможность покупать и продавать виртуальные объекты: картины, фотографии, игры, музыку.

Применяется блокчейн и в игровой индустрии. Основа рассматриваемой технологии служит для реализации GameFi-проектов (от англ. game – «игра» и finance – «финансы»), сочетающие в себе игровую механику и NFT. Это онлайн-игры, которые записывают всё происходящее в игре в транзакции на блокчейн и предоставляют возможность игрокам зарабатывать реальные деньги. Блокчейн реализует покупку и приобретение виртуальных персонажей и артефактов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Blockchain.What Is Blockchain Technology? How Does It Work? – Текст: электронный. – URL: https://builtin.com/blockchain (дата обращения: 15.04.2023).

2. Блокчейн. – Текст: электронный. – URL: https://ru.wikipedia.org/wiki/Блокчейн (дата обращения: 15.04.2023).

3. История Blockchain. – Текст: электронный. – URL: https://academy.binance.com/ru/articles/history-of-blockchain (дата обращения: 15.04.2023).

4. Что такое блокчейн, где применяется и что его ждет в будущем. – Текст: электронный. – URL: https://www.banki.ru/news/daytheme/?id=10975614 (дата обращения: 15.04.2023).

УДК 004.896

Г.А. ЕРЁМИН
Рязанский государственный радиотехнический университет
имени В.Ф.Уткина

НЕЙРОСЕТИ В СИСТЕМАХ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ

Рассматривается проблема взаимодействия нейросетей и систем автоматизированного проектирования на примере прикладной программы MATLAB.

Сегодня большую популярность получил термин нейросеть. Их внедрение происходит почти во все сферы жизни человека. Сейчас эта математическая модель, работающая по принципу систем живого организма способна решать многие задачи, и с каждым днем их структура все больше усложняется, как и решаемые ими задачи.

Возникает вопрос, возможно ли их использование в системах автоматизированного проектирования, потому что это могло бы очень сильно минимизировать траты времени на производстве, в этом и заключается актуальность выбранной темы.

Для начала стоит разобраться, что такое нейросеть и в чем же заключается принцип ее работы. Нейросеть – математическая модель, работающая по принципам нервной системы живого организма. У них нет изначально заданного алгоритма действий, поэтому их решения и результаты могут быть абсолютно спонтанными. Существует множество архитектур нейронных систем со своими классификациями.

Нейросети – это компьютерные алгоритмы, моделирующие работу мозга человека. Они используются для анализа, классификации и прогнозирования данных. Как правило, нейросеть состоит из нескольких слоев, каждый из которых представляет собой набор узлов (нейронов), связанных друг с другом. Входные данные поступают на первый слой, после чего они обрабатываются и передаются на следующий слой и так далее до тех пор, пока не будет получен конечный результат.

Обучение нейросети происходит путем подачи на вход набора обучающих данных и корректировки коэффициентов связей между нейронами до достижения наилучшего результата. Созданные нейросети могут использоваться для решения различных задач, таких как обработка изображений, распознавание речи, классификация текстов и т. д.

Современные нейросети достигают высоких результатов в решении сложных задач, что делает их незаменимыми инструментами для решения многих задач в различных областях, включая медицину, финансы, промышленность и технологии.

Системы автоматизированного проектирования используются во многих отраслях промышленности, начиная от машиностроения и заканчивая архитектурной сферой. Эти системы позволяют создавать сложные 3D-модели, соблюдать технические параметры и другие требования проектирования.

Использование нейросетей в САПР позволяет значительно улучшить процесс проектирования. Например, нейросеть может использоваться для распознавания образов и определения их соответствия требованиям заказчика. Также, нейросети могут использоваться для оптимизации параметров моделирования и для автоматического создания новых дизайнерских решений.

Одним из примеров использования нейросетей в САПР является проектирование новых систем кондиционирования воздуха. В этом случае нейросеть используется для определения оптимальных параметров системы в зависимости от климатических условий и требований к качеству воздуха. Нейросеть тестирует различные комбинации параметров и выбирает наиболее эффективный вариант.

В целом, использование нейросетей в системах автоматизированного проектирования позволяет сократить время на разработку новых продуктов и снизить стоимость производства. Это связано с тем, что нейросети позволяют автоматизировать некоторые этапы проектирования, что меньше зависит от напряженности организации в процессе работы.

Целью исследования является анализ нейронных сетей в системе производства и их использование в пакетах прикладных программ, на примере моделирования стационарного фильтра.

Проверять возможность использования нейросетей будем в пакете прикладных программ MATLAB.

В MATLAB есть множество инструментов для создания, обучения и применения нейронных сетей. MATLAB имеет встроенный инструментарий для работы с глубокими нейронными сетями, видео, изображениями и звуком.

Для создания нейронной сети в MATLAB используется функция "neural network toolbox", которая позволяет создавать нейронные сети разных типов – прямого распространения, с обратной связью и рекуррентные. Также в MATLAB доступны встроенные функции для визуализации архитектуры и процесса обучения нейронной сети.

Обучение нейронной сети в MATLAB может быть выполнено различными алгоритмами, такими как обратное распространение ошибки, обратное распространение дельты или алгоритм Левенберга-Марквардта. MATLAB также позволяет проводить finetuning, transfer learning и работу с предобученными моделями.

Применение нейронных сетей в MATLAB можно использовать для разных задач, таких как классификация, регрессионный анализ, кластеризация, обработка изображений и звука.

В целом, MATLAB предоставляет мощные инструменты для работы с нейронными сетями и позволяет решать различные задачи машинного обучения, используя нейросетевой подход.

Проблема состоит в создании сети, которая в процессе изучения описывает характеристики фильтра, а затем в качестве модели фильтра применяется ради разбора выходных сигналов при случайных значениях входных сигналов. Первая часть данной темы известна как

проблема идентификации. Рассмотрим линейный стационарный фильтр, описываемый следующим рекуррентным выражением:

$y[n] = -0.5y[n-1] + 1.5y[n-2] + r[n]$

Данный цифровой фильтр второго порядка можно также представить в виде:

$$y[n] = \frac{1}{1 + 0.5z^{-1} - 1.5z^{-2}} * r[n]$$

В системе MATLAB такой фильтр описывается М-функцией:

$Y = filter([1\ 0.5\ -1.5],1,R)$

Допустим, что на вход фильтра подается сигнал:

$r(t) = \sin(10 * \sin(t) * t)$

Зададим сигнал массивом значений с периодом дискретности 0,025 с на интервале 5 с. Входную последовательность P зададим на основе текущего и двух предшествующих значений входа X. Выведем графики входного и выходного сигналов фильтра (рисунок 1).

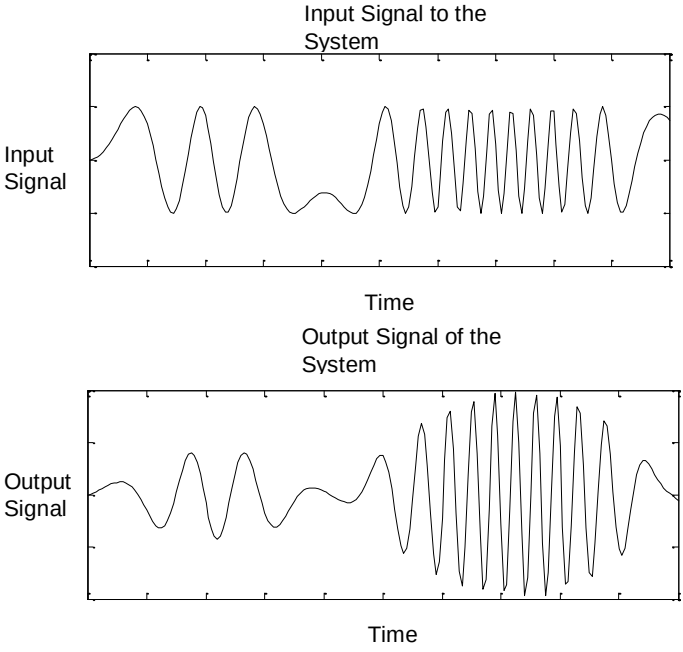


Рисунок 1 – Графики входного и выходного сигналов фильтра при моделировании фильтра средствами MATLAB

Задача нейронной сети – определить в процессе обучения параметры фильтра, а затем использовать их для моделирования при произвольных значениях входа. Вследствие того, что у такого фильтра есть один выход, то нейронная сеть должна состоять из одного нейрона. Для обработки и получения одного действующего и двух запаздывающих значений, нейрон вынужден обладать тремя входами. Пусть в качестве целевого вектора T будет массив выхода Y, тогда для входной последовательности P зададим действующее значение и два последующих значения входов R. Для получения двух запаздывающих значения, добавим функцию задержки. Получим структуру сети для моделирования фильтра, которая представлена на рисунке 2.

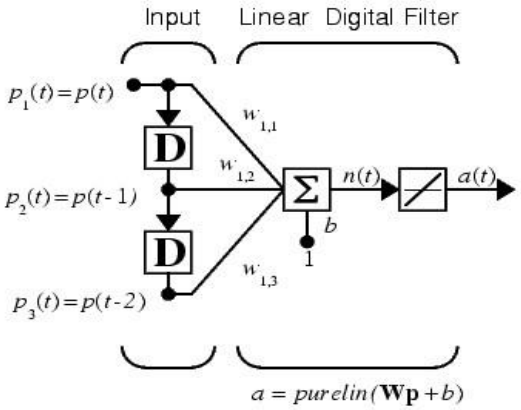


Рисунок 2 – Структура сети для моделирования фильтра второго порядка

Добавим сигналы обучения и сделаем настройку сети. Для проверки функционирования сети сравним выход a с целевой последовательностью T. Здесь же вычислим и построим график погрешности моделирования:

Результаты моделирования представлены на рисунке 3.

Результаты моделирования показывают, что сеть обеспечивает решение поставленной задачи с высокой точностью. Погрешность моделирования находится в пределах точности вычислений (10-15).

Нейросети могут использоваться для автоматизации расчёта тел сложных форм за счет предсказания объема и площади поверхности этих тел. Для этого нейросеть обучается на наборе данных, содержащем информацию о форме и размерах тел, а также их

соответствующем объеме и площади поверхности. На основе этой информации нейросеть может предсказывать объем и площадь поверхности новых тел, которые не были включены в набор данных.

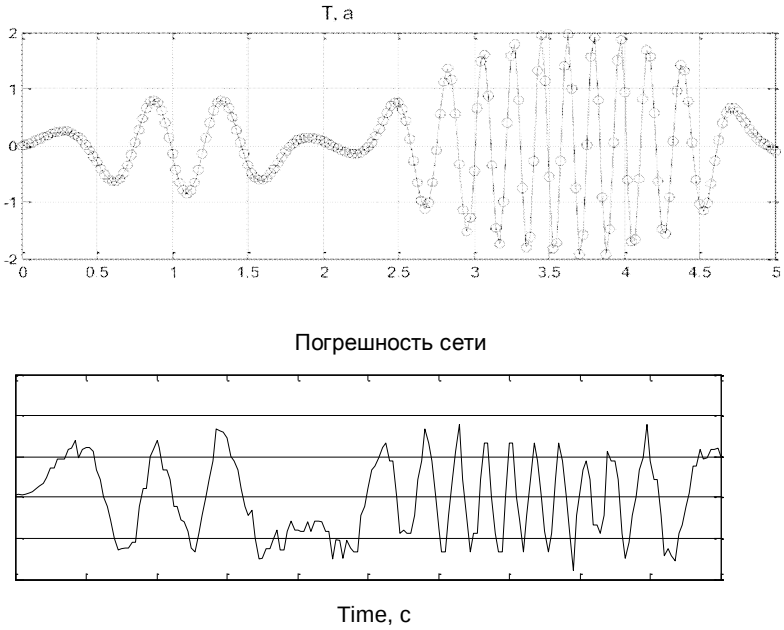


Рисунок 3 – Результаты моделирования фильтра сетью

Такой метод может быть полезен в различных областях, таких как инженерное проектирование, медицинская томография, создание компьютерных игр и виртуальной реальности. Например, нейросети могут использоваться для расчета объема мозга и других органов на основе медицинских изображений, что может помочь в диагностике и лечении различных заболеваний. В компьютерных играх и виртуальной реальности нейросети могут использоваться для автоматического расчета характеристик объектов и окружающей среды, что позволит создавать более реалистичные и интерактивные игровые миры.

Так же хочется отметить, что нейросети могут быть использованы в системах автоматизированного проектирования для обнаружения потенциальных проблем или ошибок в проектах, для улучшения точности расчетов и прогнозирования результатов, для оптимизации дизайна и ускорения процесса проектирования.

Конкретные применения нейросетей в системах автоматизированного проектирования включают:

1. Автоматическое определение геометрии объектов, таких как шестерни или корпуса, на основе входных параметров и требований.
2. Расчет механических характеристик материалов и конструкций на основе данных о физических свойствах.
3. Предсказание производительности устройств и систем на основе их параметров и условий эксплуатации.
4. Анализ данных о прошлых проектах и определение оптимальной конфигурации для новых проектов на основе этих данных.
5. Улучшение точности расчетов в системах компьютерного моделирования путем использования нейросетей для анализа больших объемов данных.
6. Автоматическая оптимизация конструкций и дизайна на основе заданных критериев, таких как минимизация веса или максимальное достижение определенной производительности.
7. Автоматическое создание и оптимизация опорных структур и балок на основе множества факторов и условий.

Использование нейросетей в системах автоматизированного проектирования может привести к более быстрым и точным результатам, более эффективному использованию ресурсов и более оптимальным дизайнам.

Использование нейросетей в системах автоматизированного проектирования имеет много перспективных возможностей. Нейросетевые методы позволяют решать сложные задачи проектирования с высокой точностью, скоростью и надежностью. Они имеют большой потенциал при оптимизации процессов проектирования и снижении издержек на разработку новых продуктов. Однако, как и во всех других технологиях, есть и недостатки. Нейросети требуют большого объема данных, вычислительных ресурсов и экспертных знаний, что может затруднить реализацию в практических приложениях. Кроме того, для каждого конкретного проекта требуется своя нейросеть, что приводит к необходимости разработки и поддержки большого числа моделей. В целом, несмотря на ограничения, нейросетевые методы в области автоматизированного проектирования могут существенно повысить эффективность и качество процесса проектирования.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Комарцова Л.Г. Нейрокомпьютеры: учебное пособие для вузов. – 2-е изд. / Л.Г. Комарцова, А.В. Максимов. – Изд-во МГТУ им. Н.Э. Баумана, 2004. – 400с.
2. Медведев В.С. Нейронные сети. MATLAB 6 / В.С. Медведев, В.Г. Потёмкин; Под общ. ред. к.т.н. В. Г. Потёмкина. – М.; ДИАЛОГ – МИФИ, 2002. – 496 с.
3. Ясницкий Л.Н. Искусственный интеллект. Элективный курс: учебное пособие / Л.Н. Ясницкий. – М.: БИНОМ, Лаборатория знаний, 2011. – 200 с.
4. Комашинский В.И. Нейронные сети и их применение в системах управления и связи / В. И. Комашинский, Д.А. Смирнов. – М.: Горячая линия. – Телеком, 2003. – 94 с.
5. Документация MATLAB. – Текст: электронный. – URL: http://matlab.exponenta.ru/neuralnetwork/ (дата обращения: 15.04.2023).

УДК 004.65

А.В. ЖИЛЬЦОВ
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ПРОЕКТИРОВАНИЕ ЭТАПОВ РАЗРАБОТКИ БАЗЫ ДАННЫХ ДЛЯ СОЗДАНИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ МОНИТОРИНГА РАЗРАБАТЫВАЕМЫХ ИЗДЕЛИЙ ПРЕДПРИЯТИЯ

Рассматривается актуальность применения баз данных на примере использования их на предприятии. Проектируются основные этапы разработки такой базы данных.

В современном мире трудно представить работу любой организации без использования баз данных (БД). Необходимо хранить большие объемы информации, и наличие такого программного средства может значительно облегчить работу всего предприятия или фирмы [1].

Используются такие БД, как правило, в составе информационной системы. Сама же информационная система состоит из комплекса программных средств, обеспечивающих работу БД, а также из людей, обслуживающих работу этой информационной системы, и ее пользователей.

На предприятии, при выпуске продукции, необходимо отслеживать каждый производственный этап. Необходимо это как минимум для того, чтобы следовать утвержденному производственному плану и не нарушать сроки изготовления продукции. Также необходимость заключается в контроле и своевременном реагировании при возникновении и соответственно решении технических вопросов, возникающих на разных этапах производства.

Таким образом, актуальность создания базы данных мониторинга разрабатываемых изделий (деталей) в производстве заключается в том, чтобы уменьшить трудоёмкость, связанную с мониторингом выпускаемой продукции и, таким образом, увеличить эффективность всего предприятия.

Для того чтобы реализовать такую базу данных необходимо определить этапы разработки создаваемого программного обеспечения.

На рисунке 1 представлены основные этапы разработки данного программного обеспечения [2].

Рисунок 1 – Этапы разработки программного обеспечения

Первым этапом является постановка задачи. В нем формулируется задача и ставится цель ее решения. Значительное внимание в ходе формулирования задачи направлено на детальное описание входной, выходной и промежуточной информации.

Этот этап является наиболее важным и ответственным, поскольку от него зависит вся последующая работа.

Основной задачей данного этапа является необходимость спроектировать такую базу данных, которая бы обеспечивала непосредственный доступ работникам предприятия к системе мониторинга и прослеживаемости разрабатываемых изделий на предприятии.

Второй этап – математическое описание поставленной задачи и выбор метода ее решения. Здесь происходит формализованное

описание задачи, а также заключаются и определяются средствами языка математики соотношения среди исходных и результирующих данных. Математическое описание задачи позволяет обеспечить точное взаимопонимание между пользователями и разработчиками программы (БД).

На данном этапе необходимо определить структурные элементы БД, такие как поле, запись, файл (таблица).

Поле – самая простая единица логической организации данных. Она идентична единице информации – реквизиту. Для того, чтобы описать поля применяются такие характеристики, как имя, тип, длина и точность числовых данных.

Под записью понимают объединение из нескольких логически взаимосвязанных полей.

Экземпляр записи – это конкретно созданная запись, содержащая определенные значения ее полей.

Файл (таблица) – объединение экземпляров записей с общей структурой. Здесь указываются поля, значения которых представляют собой определенные ключи, такие как:

- первичные (ПК или главный ключ) – устанавливают экземпляр записи;
- вторичные (ВК или внешний ключ) – исполняют роль поиска экземпляров записи.

Примером, показывающим работу данного этапа, может служить создание полей «Наименование ДСЕ», «Обозначение ДСЕ», «Состояние». Первичным ключом будет служить поле «Обозначение ДСЕ». Записью в данном случае будет: «Деталь 1», «РГРТУ.123456.789», «Изготавливается в цехе 1». Таблица будет представлять собой определенную последовательность записей, продолженная по аналогии.

Третий этап представляет собой алгоритмизацию решения поставленной задачи, а именно создание оригинального или адаптацию какого-либо известного алгоритма.

Алгоритмизация представляет собой непростой, отчасти творческий процесс. В основе его лежит понятие – алгоритм. Алгоритм – это ряд правил, которые определенным образом устанавливают последовательность исполнения операций для решения какого-либо класса задач за конкретное число шагов.

Алгоритм запуска приложения можно представить в виде простой авторизации пользователя в системе. После чего он может работать с базой данных.

Далее необходимо будет сделать запись на изготовление какого-либо изделия или входящего в него другой ДСЕ (деталь/сборочная единица).

Составление (адаптация) программ (кодирование) будет последним этапом создания программных средств. Процедура кодирования представляет собой перевод описания алгоритма на один из доступных для компьютера языков программирования.

Для того чтобы программно реализовать базу данных необходимо выбрать СУБД. Системы управления базами данных (СУБД) представляет собой программные средства, рассчитанные для создания, заполнения, обновления и/или удаления баз данных.

Предлагается разрабатывать базу данных в СУБД «1С: Предприятие». Данный продукт разработан российской компанией и уже используется в большинстве предприятий при решении подобного рода производственных задач.

Этап тестирования и отладки. Каждый из этих процессов функционально взаимосвязан друг с другом, однако их цели немного отличаются друг от друга.

Тестирование является набором определенных действий, которые позволяют продемонстрировать работу программы, а также убедиться в ее правильности выполнения.

Далее, после этапа тестирования необходимо произвести отладку. Отладка необходима для устранения ошибок в программе, начиная с момента обнаружения их в программном коде и завершая устранением причин возникновения этих ошибок.

По характеру возникновения ошибок их подразделяют на синтаксические и логические.

Далее рассматривается логика работы программы (БД) на основе изначально выданных данных. Если результат устраивает заданным требованиям, то переходят к следующему этапу.

Приемо-сдаточные испытания. На данном этапе производят приемку – передачу отлаженной базы данных вместе с сопроводительной документацией пользователю для эксплуатации. По завершении работы комиссии оформляется акт приемки-передачи.

Опытная эксплуатация. В процессе внедрения и эксплуатации прикладных программных средств могут выявляться ошибки, не обнаруженные разработчиком при тестировании и отладке программных средств. Для этого необходим данный этап. Он позволяет произвести «опытное» исследование созданной базы данных в «реальных» условиях эксплуатации.

Промышленная эксплуатация. Является завершающим этапом. Исключительно после окончания промежутка времени опытной эксплуатации и ликвидации обнаруженных на этом этапе ошибок и учета замечаний/пожеланий заказчика, происходит передача программного средства в промышленную эксплуатацию.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Управление данными. – Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова. – 2018. – Текст: электронный. – URL: https://www.voenmeh.ru/ (дата обращения: 15.04.2023).

2. Волков А.Н. Конспект лекций по дисциплине «Современные информационные технологии». – Тверь, Тверской государственный технический университет, 2012.

УДК.004.8

Е.С. ЗАЙЦЕВ
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ИССЛЕДОВАНИЕ ПРИМЕНЕНИЯ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА В МУЗЫКЕ

Рассматриваются различные способы применения искусственного интеллекта в музыкальной сфере.

Искусственный интеллект уже давно перестал быть элементом научно-фантастических фильмов и стал реальностью в нашей жизни. Он проникает во все сферы нашей деятельности, включая музыку. Сегодня искусственный интеллект используется для создания новых музыкальных произведений, анализа и классификации музыки, а также для улучшения звучания музыкальных инструментов.

Например, компания Amper Music разработала программу, которая позволяет создавать музыку на основе заданных параметров, таких как жанр, настроение и темп. Программа использует нейронные сети и алгоритмы машинного обучения для создания уникальных музыкальных треков.

Еще одним примером использования искусственного интеллекта в музыке является улучшение звучания музыкальных инструментов. Компания Stradivari Instruments разрабатывает программу, которая использует алгоритмы машинного обучения для анализа звучания

скрипок Страдивари. Эта программа может помочь мастерам в реставрации и восстановлении старинных инструментов.

Искусственный интеллект также может помочь в распространении и продвижении музыки. Например, Yandex использует алгоритмы машинного обучения для рекомендации новых треков и плейлистов на основе предпочтений слушателя.

Также он может использоваться для анализа и классификации музыки. Например, приложения для распознавания музыки, такие как Shazam и SoundHound, используют алгоритмы машинного обучения для определения названия и исполнителя трека по его звучанию.

Определение жанра

Искусственный интеллект (ИИ) уже давно используется в музыкальной индустрии для создания и производства музыки. Но недавно ИИ начал использоваться и для определения жанра музыки.

Определение жанра музыки – это важный аспект музыкальной индустрии. Жанр музыки определяет, какую аудиторию привлечет музыкальное произведение и как оно будет распространяться. Определение жанра музыки также помогает в категоризации музыкальной библиотеки и рекомендации новых песен для слушателей.

Использование ИИ для определения жанра музыки возможно благодаря большому количеству данных, которые можно обработать. ИИ может анализировать звуковые характеристики музыкальных произведений, такие как темп, ритм, мелодия и гармония, чтобы определить жанр. Кроме того, ИИ может использовать информацию о популярности и продажах альбомов в определенном жанре для более точного прогнозирования жанра новых песен.

Одна из компаний, которая использует ИИ для определения жанра музыки – это Yandex. Yandex использует ИИ для категоризации музыкальной библиотеки и создания рекомендаций для слушателей. Использование ИИ позволяет Yandex предлагать пользователю новые песни и исполнителей, которые соответствуют его музыкальным предпочтениям.

Другой пример использования ИИ для определения жанра музыки – это проект от IBM Watson. IBM Watson использует ИИ для анализа звуковых характеристик музыки и определения жанра. Этот проект может быть использован для категоризации музыкальной библиотеки и создания персонализированных рекомендаций для слушателей.

Однако, несмотря на все преимущества использования ИИ для определения жанра музыки, этот метод не является безошибочным.

ИИ может ошибаться в определении жанра, особенно если музыкальное произведение сочетает в себе элементы разных жанров или не имеет четкого определения жанра. Кроме того, ИИ не может учитывать контекст и эмоциональную составляющую музыки, что может быть важным для слушателей.

Тем не менее, использование ИИ для определения жанра музыки – это важный шаг в развитии музыкальной индустрии. ИИ может помочь в категоризации музыкальной библиотеки и создании персонализированных рекомендаций для слушателей, что повысит удобство использования музыкальных сервисов и улучшит опыт прослушивания музыки.

Для определения жанра музыки с помощью ИИ используются алгоритмы машинного обучения, такие как нейронные сети и классификаторы. Используемые данные включают звуковые характеристики музыкальных произведений, такие как темп, ритм, мелодия и гармония, а также информацию о популярности и продажах альбомов в определенном жанре. Эти данные могут быть получены из открытых источников, таких как базы данных музыкальных произведений или сервисы потоковой передачи музыки.

После обучения ИИ может использоваться для определения жанра новых музыкальных произведений. Для этого звуковые характеристики нового произведения анализируются ИИ, который определяет соответствующий жанр на основе своей обученной модели.

В целом, использование ИИ для определения жанра музыки является важным техническим достижением в музыкальной индустрии, которое позволяет улучшить категоризацию музыкальной библиотеки и создать персонализированные рекомендации для слушателей.

Альтернативное использование ИИ в музыке

Существует множество способов, которыми ИИ может быть использован в музыке. Одним из наиболее распространенных является создание музыки с помощью алгоритмов. Эти алгоритмы могут использоваться для создания мелодий, гармоний и ритмов, а также для создания текстов песен.

Другой способ, которым ИИ может быть применен в музыке – это анализ музыки. С помощью ИИ можно анализировать звуковые данные и определять характеристики музыки, такие как тональность, темп и ритм. Эта информация может быть использована для создания новых композиций и для улучшения существующих.

ИИ также может быть использован для улучшения производительности музыкальных инструментов. Например, для

создания программного обеспечения, которое помогает музыкантам играть на инструментах более точно и эффективно.

Некоторые компании уже начали применять ИИ в музыке. Например, Google создала приложение Magenta, которое использует ИИ для создания новых мелодий. IBM также разработал программное обеспечение Watson Beat, которое может помочь музыкантам создавать ритмы и гармонии.

Однако не все музыканты и любители музыки одобряют использование ИИ в музыке. Некоторые считают, что это может уменьшить значимость человеческого творчества и вмешаться в естественное развитие музыки. Однако, как и во всех областях, где применяется ИИ, это вопрос личного выбора и предпочтений.

Использование ИИ для определения жанра музыки имеет большой потенциал для развития музыкальной индустрии. Это помогает улучшить категоризацию музыкальной библиотеки, создавать персонализированные рекомендации для слушателей и облегчить поиск музыки. Однако, необходимо учитывать, что ИИ не всегда может дать точный результат, так как жанры музыки могут быть очень субъективными и различаться в зависимости от региона и культурных особенностей. Поэтому, использование ИИ для определения жанра музыки должно быть дополнено экспертным мнением и оценками слушателей.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Choi, K., Fazekas, G., Sandler, M. Automatic tagging using deep convolutional neural networks // IEEE Transactions on Multimedia. – 2017. – Vol. 19 (8)/ – Pp.1778-1788.

2. Li, X., Yang, Y. A survey on music genre recognition using machine learning // Journal of Intelligent Information Systems. – 2017. – Vol. 48 (2). Pp. 303-322.

УДК.004.421

Е.С. ЗАЙЦЕВ
Рязанский государственный радиотехнический университет
имени В.Ф.Уткина

ИССЛЕДОВАНИЕ, СРАВНИТЕЛЬНЫЙ АНАЛИЗ И РАЗРАБОТКА ВОЛНОВЫХ АЛГОРИТМОВ ТРАССИРОВКИ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ JAVA

Рассматриваются программная реализации алгоритмов трассировки Ли и их сравнительный анализ.

В настоящее время печатные платы являются основополагающими элементами любого устройства, поэтому их разработка и проектирование имеет важную роль.

Заключительным этапом конструкторской деятельности по проектированию печатных плат является трассировка.

Трассировка печатных плат заключается в определении мест расположения проводников на печатной плате вручную или с использованием одной из САПР, предназначенной для проектирования печатных плат.

Существует множество различных алгоритмов, позволяющих выполнить трассировку печатных плат. Одним из самых распространенных можно считать алгоритм волновой трассировки Ли и его модификации.

В данной статье разработана визуальная программная система на языке программирования высокого уровня Java, реализующая волновой алгоритм трассировки Ли и его модификации для их сравнительного анализа.

Алгоритм волновой трассировки Ли

В основе алгоритма Ли и его модификациях лежит разбиение монтажной плоскости на элементарные прямоугольники (рисунок 1).

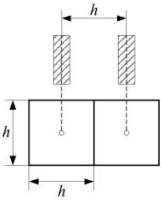


Рисунок 1 – Ячейки поля

В нашем случае, алгоритм минимизирует длину соединения. Поэтому формулу для определения веса можно представить в виде:

$$P_i = P_{i-1} + 1$$

Процесс работы алгоритма Ли показан на рисунке 2.

На данном рисунке черные ячейки – препятствия, значение по центру ячейки – значение весовой функции, значение в верхнем правом углу – номер итерации, стрелки отображают предшественника (родителя) данной ячейки, а зеленым выделен проложенный маршрут (трасса) минимальной длины.

Алгоритм кодирования веса по модулю 3

Данный алгоритм позволяет использовать меньшее количество памяти ЭВМ, поскольку для кодирования ячейки в этом методе

используется лишь три следующих состояния: свободна, занята, имеет вес (1, 2 или 3).

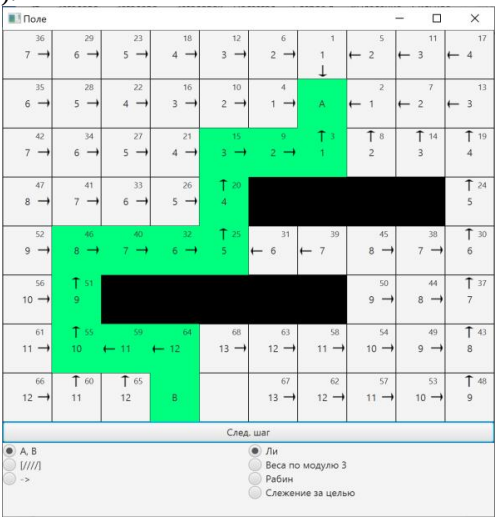


Рисунок 2 – Распространение волны для алгоритма Ли

Результат работы метода кодирования весов по модулю три показан на рисунке 3.

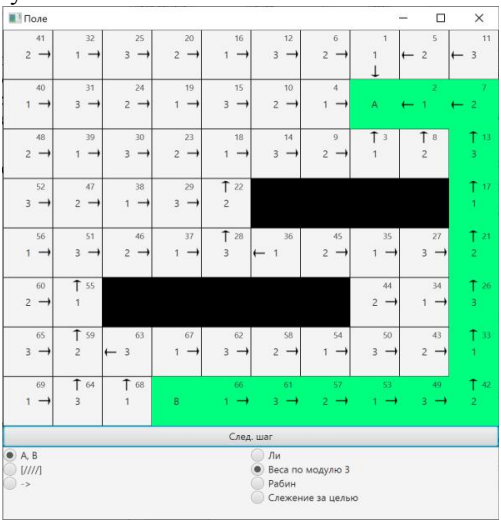


Рисунок 3 – Распространение волны алгоритма кодирования весов по модулю 3

Алгоритм Рабина

Быстродействие, которое достигается в алгоритме Рабина, осуществляется за счет приоритета распространения волны в направлении целевой ячейки.

Весовая функция для алгоритма Рабина описывается следующим выражением:

$$P_i = L(A,i) + H(i,B)$$

Результат выполнения алгоритма Рабина показан на рисунке 4.

Алгоритм слежения за целью

Сократить зону поиска можно так же при трассировке методом слежения за целью. В этом алгоритме весовая функция для ячейки:

$$H(i,B) = | x_i - x_B | + | y_i - y_B |$$

Таким образом, распространение волны осуществляется из тех ячеек фронта, которые ближе всего расположены к целевой ячейке.

Результат работы алгоритма слежения за целью показан на рисунке 5.

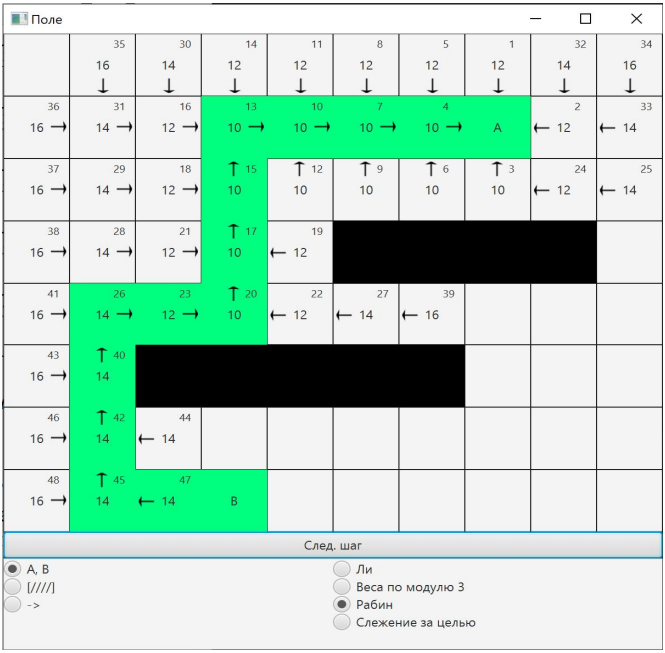


Рисунок 4 – Результат распространения волны по алгоритму Рабина

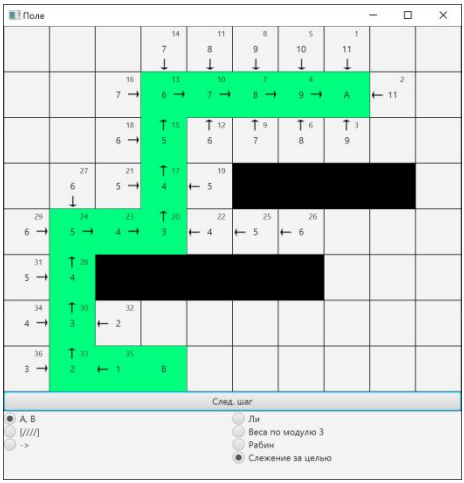


Рисунок 5 — Распространение волны по алгоритму слежения за целью

В ходе выполненной работы был создан инструмент, который позволяет выполнить трассировку волновым алгоритмом Ли и его модификациями. Так же в ходе работы был проведен сравнительный анализ данных алгоритмов, который показал, что для алгоритма Ли было необходимо просмотреть 69 ячеек, для алгоритма кодирования весов по модулю три 70 ячеек, для алгоритма Рабина 48 ячеек, а для алгоритма слежения за целью 37 ячеек. Из чего можно сделать вывод, что по скорости выигрывает алгоритм слежения за целью, но у него есть недостатки, главным из которых является потеря глобального минимума. Следовательно, оптимальным алгоритмом для решения задач трассировки можно считать алгоритм Рабина. Однако, каждый из рассмотренных алгоритмов волновой трассировки удобно использовать в подходящей для него ситуации.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Сапрыкин А.Н. Алгоритмические методы автоматизации конструирования электронных средств: учебное пособие – Рязань: ИП Коняхин А.В. (Book Jet), 2021. – 116 с.

2. Хорстманн К.С. Java. Библиотека профессионала. Том 1. Основы. – 2014 – 864 с.

УДК 004.4

Д.В. ЗАМЕШАЕВ, С.В. СКВОРЦОВ
Рязанский государственный радиотехнический университет
имени В. Ф. Уткина

АЛГОРИТМ ПОСТРОЕНИЯ УПРАВЛЯЮЩЕГО ГРАФА ПРОГРАММЫ ДЛЯ ЗАДАЧ ТЕСТИРОВАНИЯ

Рассматривается задача структурного тестирования программных модулей и предлагается алгоритм автоматического формирования управляющего графа программы на основе исходного текста, представленного на языке программирования высокого уровня.

Одним из основных этапов создания программного обеспечения является тестирование, одним из видов которого является структурное тестирование, или тестирование методом «белого ящика» [1]. Такой вид тестирования обычно применяется для анализа логики работы относительно небольших программных модулей.

Для реализации структурного тестирования требуется графовая модель программы, позволяющая построить все возможные комбинации независимых ветвей программы. Известно несколько основных видов графовых моделей программ: граф зависимостей по данным, управляющий граф, граф программных зависимостей, потоковый граф и др. [1, 2].

Граф зависимостей по данным строится на множестве вершин, соответствующих операциям над переменными, которые связываются дугами, обозначающими пары упорядоченных операций. Такой граф используется для анализа потенциального параллелизма операций программы в пределах линейных участков и не позволяет описать циклы и ветвления в программе. Эта модель используется для вычисления некоторых числовых характеристик параллельных процессов, порождаемых линейным программным сегментом.

Граф программных зависимостей представляет собой более универсальную модель программы. Вершинами в данном графе являются операторы и предикатные выражения (или операторы и операнды), а дуги, инцидентные вершине, указывают значения данных, от которых зависят операции в этой вершине, и управляющие условия, от которых зависит исполнение операций. Данная модель может отражать как зависимости по данным, так и как зависимости по управлению. Если в результате перестановки двух операторов значение некоторой переменной в одном из этих операторов может оказаться неверным, то имеется зависимость по данным. Если

существует зависимость между оператором и предикатом, значение которого непосредственно управляет исполнением оператора, то это зависимость по управлению.

Управляющий граф представляет собой подграф графа программных зависимостей, характеризующийся следующими особенностями. Это ориентированный граф с единственной входной вершиной (далее обозначается как *s*), и единственной выходной вершиной (далее обозначается как *t*). Каждая вершина в графе имеет не больше двух потомков. Дугами обозначается поток управления. Все вершины управляющего графа разделяются на предикатные и операторные. Предикатные вершины соответствуют простым условиям в программе. Сложные условия представляются в виде совокупности простых. Операторным вершинам соответствуют линейные участки программы (от одного до нескольких операторов программы). Предполагается, что для любой вершины *N* в данном графе существует путь из *s* в *N*, и путь из *N* в *t*.

Потоковый граф программы подобен управляющему графу, но ориентирован на решение задач структурного тестирования программ. Поэтому рассмотрим процесс его построения и укажем некоторые особенности на примере программы, показанной на рисунке 1.

```
1  Program prg1;
2  var
3    i, n, f: integer;
4  Begin
5    write('Введите число: ');
6    read(n);
7
8    f := 1;
9    for i:= 0 to n do
10     f *= n;
11
12   write('Полученный факториал: ', f);
13 End.
```

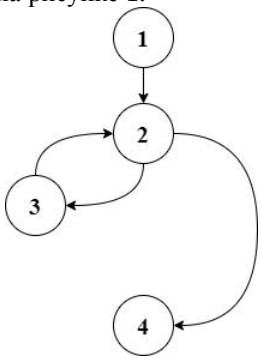


Рисунок 1 – Текст программы и потоковый граф

Начальной вершине *s* сопоставляется строка 1 программы, конечной вершине *t* соответствует строка 13. Строки 2 – 8 описывают линейную последовательность операторов и их можно объединить с начальной вершиной *s*. Заголовок цикла в строке 9, как и любой цикл со счетчиком, отображается следующими вершинами графа:

- операторный узел, задающий начальное значение счётчика цикла (в примере этому узлу соответствует оператор i := 0);

- предикатный узел, описывающий условие выхода из цикла (в примере он задает условие i < n);
- подмножество узлов, соответствующих телу цикла;
- операторный узел, обеспечивающий инкремент счётчика цикла (в примере для цикла for такому узлу соответствует строка i to n).

Особенностью отображения циклов является то, что выход из цикла, как и вход в него, осуществляется только через заголовок. Поэтому последний операторный узел (инкремент) связывается дугой с предикатным узлом (условие выхода из цикла).

Потоковый граф программы, показанный на рисунке 1, имеет следующую семантику вершин:

- вершина 1 соответствует строкам программы 1 – 8, включая операторный узел, задающий начальное значение параметра цикла;
- вершина 2 – предикатный узел цикла (строка 9);
- вершина 3 – тело цикла (строка 10) и инкремент счётчика цикла (строка 9);
- вершина 4 – строки программы 11 – 13.

Циклы с предусловием в потоковом графе отображаются аналогично, за исключением того, что не выделяются операторные узлы, связанные со счётчиком, ввиду его отсутствия.

Циклы с постусловием не имеют операторные узлы, связанные со счётчиком. Вход в такой цикл осуществляется через операторный узел, соответствующий ключевому слову repeat, а выход - через предикатный узел, задающий условие выхода из цикла (ключевое слово until). Одна из дуг этого предикатного узла направлена в операторный узел, соответствующий ключевому слову repeat, а вторая дуга – в следующий узел.

Процесс разбиения сложного условия на множество простых, можно показать на примере фрагмента программного кода, показанного на рисунке 2.

```
1  if a < b and b = 10 then x
2  else y
3  write('...');
```

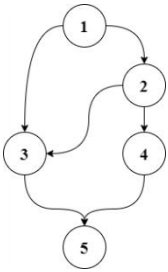


Рисунок 2 – Отображение кода сложного условия

Сложное условие (рисунок 2) можно разбить на два простых: a < b и b = 10. Разбиение осуществляется по логическим операторам and и or. Полученный граф имеет следующую семантику вершин: вершина 1 соответствует условию a < b; вершина 2 определяет условие b = 10; вершина 3 отображает оператор else y; вершина 4 – then x; вершина 5 – write('…').

Полученный потоковый граф программного модуля позволяет решить конкретные задачи структурного тестирования. Например, можно вычислить цикломатическую сложность, которая показывает количество независимых путей и дает оценку количества тестов, гарантирующих однократное выполнение всех операторов. Для формирования независимых путей можно использовать алгоритм, предложенный в работе [3].

В заключение следует отметить, что графовые модели программ используются не только в задачах тестирования. Они широко применяются при оптимизации программного кода, организации параллельных вычислений и в других прикладных задачах [4].

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Орлов С.А., Цилькер Б.Я. Технологии разработки программного обеспечения. – СПб.: Питер, 2021. – 608 с.

2. Векторизация программ: теория, методы, реализация: сб. статей. – М.: Мир, 1991. – 275 с.

3. Рудаков В.Е., Скворцов С.В. Построение базового множества независимых путей потокового графа для тестирования программных модулей // Системы управления и информационные технологии. – 2012. – Т. 50. – № 4. – С. 67-70.

4. Касьянов В.Н., Евстигнеев В.А. Графы в программировании: обработка, визуализация и применение. – СПб.: БХВ-Петербург, 2003. – 1104 с.

УДК 004.65

М.И. КИРЬЯНОВ

Рязанский государственный радиотехнический университет имени В.Ф. Уткина

РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ УЧЕТА ПРОЕКТНЫХ РЕШЕНИЙ ДЛЯ ИХ ЭКСПЕРТНОЙ ОЦЕНКИ

Рассматривается процесс разработки информационной системы учета проектных решений для их экспертной оценки и предоставляется примерная структура базы данных для этой системы.

В современном мире разработка программного обеспечения является одной из ключевых отраслей IT-сферы. Каждый год компании тратят миллионы долларов на разработку и совершенствование своих программных продуктов. Однако, не все проекты достигают своих целей и приносят желаемую прибыль. Одним из факторов, влияющих на успех проекта, является качество принимаемых решений на различных этапах разработки. Поэтому оценка проектных решений является важным этапом в процессе разработки любого программного проекта. Оценка позволяет выявить и устранить ошибки в проекте на ранних стадиях, что значительно снижает вероятность неудачной реализации проекта.

Для автоматизации процесса оценки проектных решений можно использовать информационную систему. Разработка такой системы позволит оптимизировать процесс оценки, повысить его точность и улучшить качество проектов.

Целью данной статьи является рассмотрение процесса разработки информационной системы учета проектных решений для их экспертной оценки и предоставление примерной структуры базы данных для этой системы.

Процесс разработки информационной системы включает в себя несколько этапов:

1. Анализ требований.

Первый шаг в разработке информационной системы учета проектных решений для их экспертной оценки — это анализ требований. На этом этапе необходимо выяснить, какие функции должна выполнять система, какие данные она должна хранить и как пользователи будут взаимодействовать с системой. Также на этом этапе определяются технические требования к системе.

2. Проектирование структуры базы данных.

После анализа требований необходимо приступить к проектированию структуры базы данных. Основными таблицами базы данных в данной системе будут являться таблицы с информацией о проектных решениях и таблицы с информацией об экспертной оценке. Каждая таблица будет содержать набор полей, определяющих характеристики записей в этой таблице.

3. Разработка пользовательского интерфейса.

Пользовательский интерфейс – это то, как пользователи будут взаимодействовать с системой. В данной системе пользователи будут иметь доступ к списку проектных решений, которые нужно оценить, и к списку экспертов, которые будут проводить оценку. Также

пользователи будут иметь возможность просматривать результаты оценки проектных решений.

4. Разработка бизнес-логики.

Бизнес-логика – это набор правил и процедур, которые определяют, как система будет работать. В данной системе бизнес-логика будет включать процедуры получения данных о проектных решениях и экспертной оценке, а также процедуры распределения проектных решений между экспертами.

5. Тестирование и отладка.

Последний этап в разработке информационной системы – это тестирование и отладка. На этом этапе необходимо проверить работоспособность системы и исправить возможные ошибки.

Проблемы, возникающие при разработке информационной системы учета проектных решений для их экспертной оценки, включают в себя:

- определение списка критериев оценки проектных решений, который должен быть составлен в соответствии с конкретной областью деятельности;
- создание системы сбора данных о проектных решениях и экспертной оценке, которая должна быть удобной и доступной для использования;
- создание системы сбора данных о проектных решениях и экспертной оценке, которая должна быть удобной и доступной для использования;
- организация процесса экспертной оценки проектных решений, который должен проводиться с помощью определенного набора инструментов и методов;
- анализ полученных результатов экспертной оценки и создание соответствующих отчетов и аналитических материалов.

Информационная система должна обеспечивать следующую функциональность:

1. Создание, редактирование и удаление проектов и задач.

С помощью информационной системы пользователи должны иметь возможность создавать, редактировать и удалять информацию о проектах и задачах для каждого проекта. Для создания проекта необходимо заполнить форму с информацией о проекте, а для создания задачи – форму с информацией о задаче и ссылкой на проект, к которому она относится. Для редактирования проекта или задачи необходимо выбрать соответствующий проект или задачу из списка и внести необходимые изменения. Для удаления проекта или задачи

необходимо выбрать соответствующий проект или задачу из списка и подтвердить удаление.

2. Назначение ответственного за проект.

Пользователи должны иметь возможность назначать ответственного за выполнение проекта. Для этого необходимо выбрать соответствующий проект из списка и назначить сотрудника, который будет отвечать за выполнение проекта.

3. Добавление, редактирование и удаление экспертов.

С помощью информационной системы пользователи должны иметь возможность добавлять, редактировать и удалять информацию об экспертов. Для добавления эксперта в систему необходимо заполнить форму с информацией об эксперте. Для редактирования информации необходимо выбрать соответствующую запись об эксперте из списка и внести необходимые изменения. Для удаления информации об эксперте необходимо выбрать соответствующую запись из списка и подтвердить удаление.

4. Экспертная оценка проектных решений.

Пользователи должны иметь возможность запрашивать экспертную оценку проектных решений. Для этого необходимо создать решение и выбрать эксперта, который даст оценку данному решению. Эксперт должен иметь возможность просмотреть информацию о проекте и оценить решение с помощью системы.

5. Просмотр и фильтрация списка проектов и решений.

Пользователи должны иметь возможность просмотреть список проектов и решений и отфильтровать их по различным параметрам, таким как статус, дата, ответственный, эксперт и т.д.

6. Генерация отчетов.

С помощью информационной системы пользователи должны иметь возможность генерировать отчеты по проектам и решениям, включающие информацию о статусе проектов, выполненных задачах, экспертных оценках и т.д.

Информационная система учета проектных решений должна иметь определенную структуру базы данных, которая должна быть спроектирована таким образом, чтобы обеспечить эффективную работу системы и хранение всех необходимых данных. Ниже приведена примерная структура базы данных для такой системы.

- Таблица «Эксперты» – содержит информацию об экспертах, которые будут оценивать проектные решения. Для каждого эксперта в таблице сохраняются уникальный идентификатор, имя, контактные данные (номер телефона, электронная почта и т.д.) и другие дополнительные сведения.

- Таблица «Проекты» – содержит информацию о каждом проекте, включая уникальный идентификатор проекта, наименование проекта, описание проекта, дату создания проекта, статус проекта (например, «завершен», «в процессе», «в ожидании») и т.д.
- Таблица «Оценки» – содержит информацию об оценках, которые были даны экспертами для каждого проекта. Для каждой оценки сохраняется уникальный идентификатор, идентификатор проекта, идентификатор эксперта, дата оценки, оценка проекта (например, от 1 до 10) и комментарии.
- Таблица «Права доступа» – содержит информацию о пользователях системы, их логин или электронную почту и пароль для входа в систему, и роль пользователя в системе (например, администратор, эксперт).

Приведенные выше таблицы являются основными. Их количество можно увеличить, введя возможность объединения экспертов в группы, тогда в базе данных необходимо создать следующие таблицы:

- таблицу «Группа», содержащую уникальный идентификатор, информацию о группе, такую как руководитель, список участников, какого плана проектами группа будет заниматься;
- таблицу «Группа/Проект», содержащую информацию об оценках, которые были даны экспертами группы для каждого проекта;
- таблицу «Группа/Эксперт», содержащую информацию о группах и включенных в них экспертах.

Также в базу данных можно добавить дополнительные таблицы для хранения информации о файлах, связанных с каждым проектом, категории проектных решений, организациях и проектировщиках проектных решений, задачах, которые необходимо выполнить для каждого проекта, комментариях экспертов, типах оценок, используемых при оценивании и т.д. Каждая таблица может быть связана с другими таблицами через уникальный идентификатор, такой как номер проекта или номер эксперта, что позволяет быстро и эффективно находить, и анализировать информацию.

В качестве примера можно привести следующую диаграмму базы данных, приведенную на рисунке 1.

Информационная система может быть выполнена с помощью современных технологий разработки ПО, таких как Java, Python, C# и других. Кроме того, для разработки такой системы необходимо использовать современные технологии баз данных, такие как MySQL, PostgreSQL или MongoDB.

Одним из примеров информационной системы учета проектных решений в области программирования для их экспертной оценки может быть система, разработанная на базе веб-технологий. В такой системе пользователи могут работать с системой через веб-интерфейс, что позволяет им использовать систему из любого места, где есть доступ в интернет.

Разработка информационной системы учета проектных решений для их экспертной оценки – это сложный и многогранный процесс. Для успешной реализации такой системы необходимо провести анализ требований, разработать структуру базы данных и реализовать функциональность, учитывая специфику сферы применения и потребности конечных пользователей.

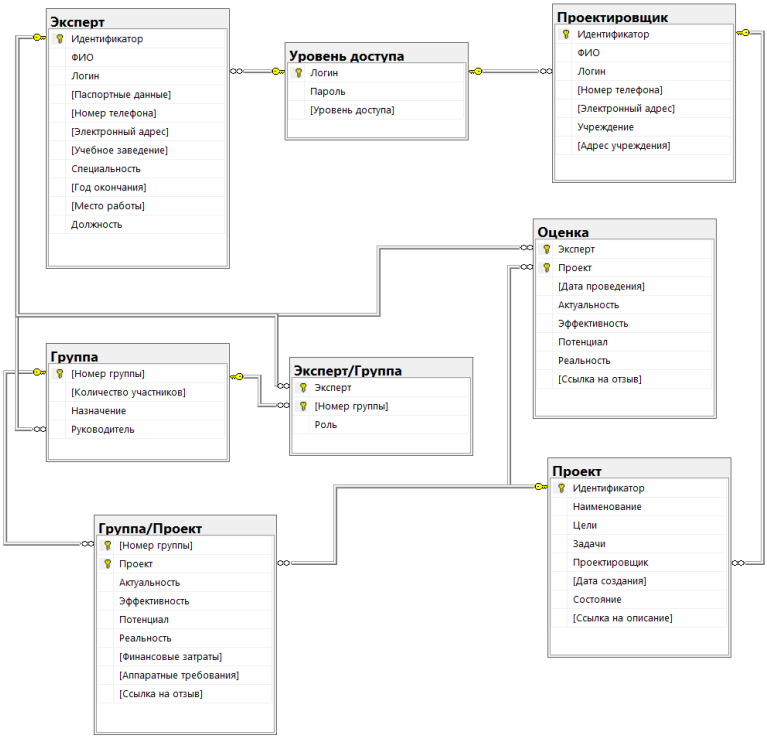


Рисунок 1 – Диаграмма базы данных для информационной системы учета проектных решений

Ключевыми преимуществами такой системы являются:

1. Удобство и простота использования. Система позволяет легко отслеживать проектные решения и оценки, а также быстро получать необходимую информацию.

2. Эффективность. Система ускоряет процесс принятия решений и позволяет проводить экспертную оценку проектов более точно и качественно.

3. Безопасность. Система обеспечивает защиту конфиденциальных данных и контроль доступа к информации.

Разработка информационной системы учета проектных решений для их экспертной оценки является важным шагом в развитии IT-сферы и других отраслей, где требуется принятие важных бизнес-решений.

Таким образом, разработка информационной системы учета проектных решений в области программирования для их экспертной оценки является важной задачей для улучшения качества принимаемых решений на различных этапах разработки программного обеспечения. Такая система позволяет учитывать все принятые решения, проводить их экспертную оценку и анализировать данные для принятия наиболее эффективных решений.

Компании и организации, внедрившие такую систему, смогут значительно повысить эффективность своей работы и получить конкурентное преимущество на рынке.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Орлов А.И. Организационно-экономическое моделирование: учебник в 3 ч. Ч. 2: Экспертные оценки. // М.: Изд-во МГТУ им. Н.Э. Баумана, 2011 г.

2. Данелян Т.Я. Формальные методы экспертных оценок // Статистика и Экономика. – 2015. – Вып. 1. – С. 183-187. – Текст: электронный. – URL: https://doi.org/10.21686/2500-3925-2015-1-183-187 (дата обращения: 15.04.2023).

3. Schwalbe K. Information Technology Project Management // Cengage Learning, 2013.

4. Groff J., Weinberg P., Oppel A. SQL: The Complete Reference, 3rd Edition // McGraw Hill, 2009.

УДК 004.415.2.031.45

Д.Ю. КОЗЫРЕВ
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

РАЗРАБОТКА АЛГОРИТМОВ ДЛЯ ДИАГНОСТИКИ ЭМОЦИОНАЛЬНОГО СОСТОЯНИЯ ЧЕЛОВЕКА

В статье рассматривается разработка нового программного продукта для ускоренного тестирования и комплексной оценки уровня психоэмоционального состояния человека

В современном мире IT-технологии проникают во все сферы жизни человека. Такие технологии используются в областях развлечений и в сферах безопасности. Также могут встречаться в медицине и в психологии. Изучение поведения человека представляет особый интерес для ученых, изучающих эмоции людей.

Цель данной работы – внедрить и протестировать программный продукт по выбранному алгоритму для оценки психоэмоционального состояния человека

Реализация программного обеспечения

Реализация поставленных задач требует, прежде всего, правильной постановки перед монитором и регулирования условий эксперимента для достижения наиболее точных результатов тестирования программы

Рисунок 1 подробно иллюстрирует расположение человека перед компьютером. Вы должны сидеть перед монитором с диагональю 15,5 дюймов, с расстоянием просмотра 50-70 см [1]. Это расстояние позволяет разместить изображение на экране полностью в поле зрения. Если зрение вам не даёт выдерживать расстояние, то следует уменьшить разрешение изображения или отрегулировать расстояние между человеком и монитором вручную.

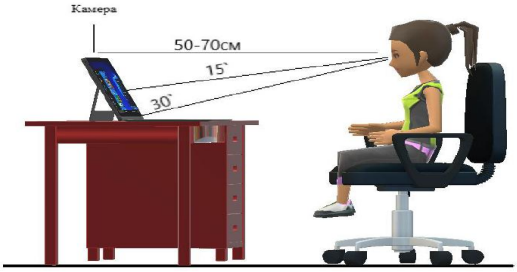


Рисунок 1 – Схема размещения перед монитором

Угол зрения лица человека, сидящего прямо, должен быть максимально прямым. Объекты, мешающие обзору, должны быть исключены из поля зрения камеры. Всё лицо должно быть освещено равномерно, при этом не допуская резких теней.

На экране в определенное время появляются различные изображения. В качестве начальных стимулов [2] будут использованы цветные изображения с эмоциями, представленные на рисунке 2, – радость, удивление, презрение и т.д.

Для создания программы был выбран язык Python. Его концепции дизайна делают работу с кодом простой и удобной, а синтаксис позволяет программистам использовать меньше строк кода, чем в таких языках, как C++ или Java. Он также включает в себя обширную стандартную библиотеку.

Рисунок 2 – База эмоций из Radboud Faces Database

Поскольку он является кроссплатформенным и имеет простую и довольно эффективную среду разработки IDLE, Python имеет значительное преимущество у программистов перед другими языками.

Для работы с техническим зрением была выбрана библиотека OpenCV [3] (Open Source Computer Vision), которая имеет более 500 функций для распознавания эмоций на лице. Также она будет полезна для отслеживания прогресса в реальном времени. Библиотека призвана предоставить базовые инструменты для решения задач компьютерного зрения и имеет большое количество плагинов для разных задач. Библиотека свободно распространяется, проста в использовании, написана на C и Python и работает практически на всех операционных системах. Помимо OpenCV была также использована библиотека для работы с техническим зрением – Dlib, которая содержит некоторые элементы, не реализованные в OpenCV.

Программа обучения нейросети Trainer

В проекте реализованы два основных скрипта – Trainer и Detector.

Trainer

Внутри скрипта для работы с техническим зрением существует несколько библиотек. Это библиотеки, которые поддерживают матрицы и многомерные массивы, также библиотека для решения очень быстрых математических функций и специальная библиотека tensorflow для работы с нейронными сетями на основе матриц. Программа получает на вход созданный блок изображения и преобразует его в матрицу (рисунок 3).

```
def full_layer(input, size):
    insize = int(input.get_shape()[1])
    W = weight_variable([insize, size])
    b = bias_variable([size])

    return tf.matmul(input, W) + b
```

Рисунок 3 – Создание матрицы

Далее на основе полученных изображений создаются различные матрицы, где каждая ячейка раскрывает свои свойства и функции для конкретного загруженного изображения (рисунок 4).

```
def test(emoji_data, sess):
    logger.info('CALCULATING TESTSET ACCURACY ...')
    L = len(emoji_data.test.labels)

    x = emoji_data.test.images
    y = emoji_data.test.labels

    accs = []

    for i in tqdm.tqdm(range(0, L, 30)):
        if i+30 <= L:
            x_i = x[i:i+30].reshape(30, 48, 48, 1)
            y_i = y[i:i+30].reshape(30, len(EMOTION_MAP))
        else:
            x_i = x[i:].reshape(L-i, 48, 48, 1)
            y_i = y[i:].reshape(L-i, len(EMOTION_MAP))

        accs.append(sess.run(accuracy, feed_dict={X:x_i, Y:y_i, keep_prob:1.0}))

    acc = np.mean(accs)
```

Рисунок 4 – Создание матриц с критическими параметрами

Detector

На данном этапе программа пытается найти на изображении с веб-камеры лицо человека. Далее нейросеть переводит найденное лицо в матрицу и сравнивает ее с паттернами, которые были заранее обучены. Затем программа делает вывод и определяет эмоцию, которая соответствует определенной матрице текущего лица. Работа программы происходит циклично до тех пор, пока камера захватывает лицо (рисунок 5).

Далее происходит обучение нейросети для создания моделей (рисунок 6).

По завершению обучения программы результаты работы (вид эмоции) записываются в Exel файл.

```
def from_cam(sess):
    start_time = int(time.time())
    counter = 1
    face_cascade = cv2.CascadeClassifier(config_parser['OPEN_CV']['cascade_classifier_path'])
    cap = cv2.VideoCapture(0)
    while True:
        pixmap = QPixmap(f'pictures/{counter}.png')
        window.label_2.setPixmap(pixmap)
        window.label_2.resize(pixmap.width(), pixmap.height())
        ret, frame = cap.read()
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, 1.3, 5)
        for (x, y, w, h) in faces:
            cv2.rectangle(gray, (x, y), (x + w, y + h), (255, 0, 0), 2)
            face_img_gray = gray[y:y + h, x:x + w]
            face_img_gray = cv2.resize(face_img_gray, (48, 48))
            s = time.time()
            emotion, confidence = inference(sess, face_img_gray)
            logger.critical('model inference time: {}'.format(time.time() - s))
            with open(f'{name}.csv', 'a') as n:
                n.write(f"{counter};{emotion};{confidence};{time.time()}\n")
            if int(time.time()) - start_time > TIME_TO_CHANGE_PICTURE:
                start_time = int(time.time())
                counter += 1
        if counter > 6:
            break
    cap.release()
```

Рисунок 5 – Основной цикл from cam – обработка текущего изображения нейросетью

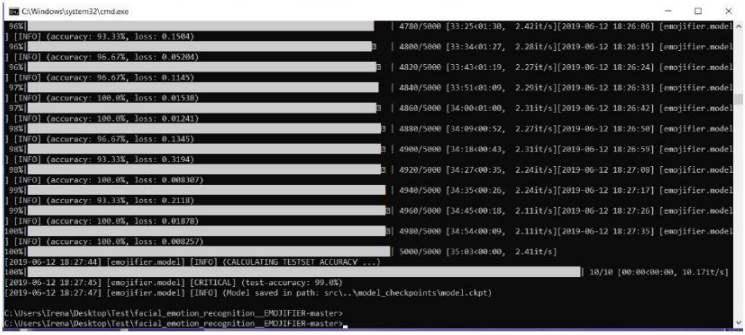


Рисунок 6 – Обучение нейросети

Таким образом был реализован алгоритм распознавания эмоций и разработан программный продукт, который позволит в последующем проводить тестирование для оценки уровня психоэмоционального состояния человека.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Эргономика рабочего места. – Текст: электронный. – URL: http://www.up-pro.ru/ergonomika-rabochego-mesta.html (дата обращения: 15.04.2023).

2. Radboud Faces Database. – Текст: электронный. – URL: http://www.socsci.ru.nl:8180/RaFD2/RaFD (дата обращения: 15.04.2023).

3. С.В. Томилов. Распознавание лиц на базе библиотеки OpenCV. – Текст: электронный. – URL: https://dspace.susu.ru/xmlui/bitstream/handle/0001.74/1504/45.pdf?sequence=1?sequence=1 (дата обращения: 15.04.2023).

УДК 004.032.26

М.А. КОМАРОВА
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

АЛГОРИТМ ВОССТАНОВЛЕНИЯ СКРЫТЫХ ЗНАЧЕНИЙ ЗАДАННОЙ РАЗМЕРНОСТИ ПО ЦИФРОВЫМ СНИМКАМ

Рассматривается разработка программного модуля информационной системы, главным функционалом которого является выявление ФИО владельца транспортного средства по имеющимся нечетким данным об автомобиле: цвете, марке и известной части госномера, извлеченных с фотографии.

С целью реализации данного алгоритма был придуман и написан программный модуль информационной системы с использованием высокоуровневого языка программирования общего назначения Python.

Перейдем к алгоритму работы программного модуля.

Шаг 1. Получение исходных данных.

Существует два метода формирования исходных данных в программном модуле:

1. В первом варианте исходные данные заносятся вручную через клавиатуру.

2. Во втором варианте одно из исходных данных, а именно известная часть регистрационного номера, распознается с фотографии, загруженной в программу, благодаря использованию библиотеки OpenCV, которая направленна на компьютерное зрение в реальном времени, а также библиотеки глубокого обучения, обеспечивающей взаимодействие с искусственными нейронными сетями – Keras, свёрточной нейронной сети и др.

Если исходные данные были получены путем ввода с клавиатуры (первый способ), то происходит переход к шагу 7. В случае, если данные были получены изображением и при помощи

программных библиотек (второй способ), то происходит переход к шагу 2.

Шаг 2. Загрузка и обработка исходного изображения (Рисунки 1-2).

Распознавание автомобильного номера проводится следующим образом: на вход поступает картинка, на ней находятся регистрационные знаки. Поиск автомобильных номеров осуществляется с помощью каскадов Хаара. В этом участвует библиотека OpenCV. При дальнейшей разработке планируется заменить этот механизм на более совершенный.

После нахождения регистрационных знаков с каждым из них индивидуально выполняется следующая последовательность действий:

1. Адаптивный пороговый анализ изображения.
2. Морфологические преобразования.
3. Аффинные преобразования.
4. При необходимости, медианное размытие.

На выходе получаем изображение как на рисунке 2.

Рисунок 1 – Данное изображение

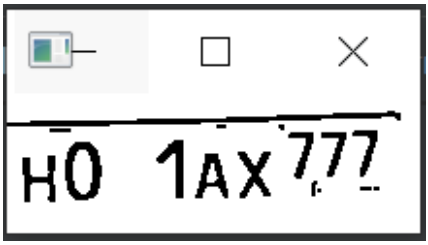


Рисунок 2 – Измененное изображение

Шаг 3. Процесс разделения цифр и букв номера на отдельные символы.

Данный процесс происходит следующим образом: выделяются контуры достаточного размера и находящиеся на нужном уровне иерархии контуров. Контуры с таким набором параметров вероятнее

всего являются контурами символов, а именно их мы и ищем (рисунок 3).

Рисунок 3 – Процесс разделения цифр и букв номера на отдельные символы

Шаг 4. Определение того, является ли номер подозрительным.

Для этого нужно посчитать среднеквадратическое отклонение расстояния между символами на регистрационном знаке. Если оно много больше единицы, то делаем предположение, что номер подозрительный. Переходим к шагу 5. Если номер не является подозрительным – завершение алгоритма.

Шаг 5. Получение известной части регистрационного номера.

Чтобы правильно распознать символы с фотографии необходимо использование свёрточной нейронной сети. В настоящее время используется нейронная сеть, обученная на основе набора данных EMNIST, созданного Лабораторией Информационных Технологий Национального Института Стандартов и Технологий США. Используя нейронную сеть и настраивая различные характеристики, мы получаем известную часть регистрационного номера (рисунок 4).

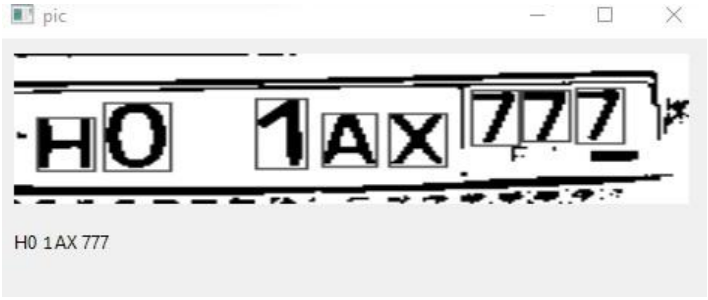


Рисунок 4 – Вывод известной части регистрационного номера

Шаг 6. Внесение оставшихся данных.

Марка и цвет автомобиля вносятся с клавиатуры.

Шаг 7. Работа с базами данных.

После получения исходных данных с помощью стороннего модуля *pyodbc* происходит установление соединения с базами данных

(БД). Для проверки работоспособности программного модуля необходимы две БД: аналог БД платного парковочного пространства и аналог БД ГИБДД. Альтернативные базы данных разработаны с применением системы управления реляционными базами данных MS SQL Server.

Для начала нам необходимо обратиться к БД ГИБДД для нахождения недостающей части госномера. Для доступа к информации о номере транспортного средства (ТС) необходимо получить доступ к полям в базе данных. Это осуществляется запросами SQL. Далее необходимо разделить *split* методом записи номера на отдельные цифры. Затем происходит определение положения нехватающих символов. По завершении данной процедуры будет создан список транспортных средств, соответствующих исходным данным. Далее необходимо произвести идентификацию автовладельца по дополнительным признакам, которые мы ввели заранее: марке и цвете автомобиля. После сопоставления исходных данных с записями в таблице базы данных находится нужная строка, из которой программа извлекает ФИО владельца автомобиля и полный регистрационный номер.

Далее полученный регистрационный номер необходимо сравнить со строками из БД платного парковочного пространства, чтобы удостовериться в том, что владелец действительно не оплатил парковку. И только после получения удовлетворительного результата можно составлять заявление на выписку штрафа за неуплату парковочного места.

Созданный программный модуль позволяет идентифицировать транспортное средство, даже если его регистрационный номер не полный. Это делает модуль очень востребованным среди владельцев парковочных пространств, которые терпят убытки из-за недобросовестных владельцев автомобилей.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Молинаро, Э. SQL. Сборник рецептов. – Пер. с англ. – СПб: Символ-Плюс, 2009. – 672 с.

2. Хеллман, Д. Стандартная библиотека Python 3: справочник с примерами, 2-е изд.: Пер. с англ. – СПб.: ООО “Диалектика”, 2019. – 1376 с.

3. Введение в OpenCV – библиотеку компьютерного зрения на Python. – Текст: электронный. – URL: https://pythonist.ru/vvedenie-v-opencv-biblioteku-kompyuternogo-zreniya-na-python (дата обращения: 15.04.2023).

4. Библиотека Keras – Русскоязычная документация Keras. – Текст: электронный. – URL: https://ru-keras.com (дата обращения: 15.04.2023).

УДК 004.032.26

М.А. КОМАРОВА
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ИСПОЛЬЗОВАНИЕ НЕЙРОННОЙ СЕТИ ДЛЯ ДООПРЕДЕЛЕНИЯ ИНФОРМАЦИИ ПО НЕЧЕТКИМ ИСХОДНЫМ ДАННЫМ

Рассматривается использование нейронной сети Keras для доопределения серии и номера паспорта, извлеченных с фотографии, при деформации.

Получение информации о серии и номере паспорта является одним из важных шагов процесса идентификации человека. Однако, когда паспорт подвергается воздействию воды, номер и серия могут исказиться и стать нечитаемыми.

Одним из способов доопределения информации является использование компьютерного зрения и нейронных сетей. Наиболее эффективным решением в этом случае будет использование языка Python с библиотекой Keras, которая позволяет строить и обучать нейронные сети.

Рассмотрим алгоритм доопределения серии и номера паспорта, извлеченных с фотографии, при деформации.

Шаг 1. Сбор и подготовка данных.

Сначала мы собираем данные для обучения нашей нейронной сети. Мы будем использовать фотографии паспорта, на которых номер и серия паспорта затруднительно читаемы из-за пролития воды.

Затем, необходимо подготовить данные – извлечь области, в которых располагаются номер и серия паспорта, обрезать их и сохранить в качестве отдельных изображений. Наша цель – удалить изображение фона и сохранить только текст на паспорте. Мы будем использовать библиотеку OpenCV для выполнения этой части работы.

Для этого можно использовать библиотеку OpenCV в Python.

С помощью OpenCV мы сможем произвести следующие операции:

- превратить цветное изображение в оттенки серого;
- очистить шумы с помощью операции фильтрации;

- применить метод бинаризации для разделения текста от фона;
- найти контуры текста на паспорте и вырезать их изображения.

Шаг 2. Обучение нейронной сети.

Далее, мы переходим к обучению нейронной сети. Для этого мы будем использовать фреймворк Keras – одну из самых популярных библиотек для построения нейронных сетей на Python.

Первым шагом будет загрузка изображений и подготовка их для обучения. Для этого мы разделим изображения на две категории: номер и серия паспорта, и будем использовать метки (номера и серии), чтобы обучать нашу нейронную сеть.

Далее, мы создадим модель сверточной нейронной сети (Convolutional Neural Network, CNN), которая будет обучаться на этих данных.

Сверточная нейронная сеть – это класс алгоритмов машинного обучения, используемых для обработки и классификации многомерных данных, таких как изображения, видео и звук. Они используют сверточные слои для поиска определенных признаков во входных данных, которые затем объединяются в более высокоуровневые признаки для более точной идентификации объектов на изображениях. Сверточные нейронные сети широко используются в различных областях, включая распознавание образов, обработку естественного языка, диагностику медицинских изображений, автоматическую обработку видео и многое другое.

Мы будем использовать архитектуру с несколькими сверточными слоями, популярными функциями активации (ReLU и Softmax), а также слоем Flatten для «распрямления» результатов свертки перед передачей в полносвязные слои.

Примерный код для обучения нейронной сети:

```
mport keras
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
# создаем модель нейронной сети
model = Sequential()
# добавляем сверточный слой
model.add(Conv2D(32, (3, 3), input_shape=(64, 64, 3), activation='relu'))
# добавляем pooling слой
model.add(MaxPooling2D(pool_size=(2, 2)))
# добавляем сверточный слой
model.add(Conv2D(64, (3, 3), activation='relu'))
# добавляем pooling слой
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
    # добавляем flatten слой
    model.add(Flatten())
    # добавляем полносвязный слой
    model.add(Dense(units=128, activation='relu'))
    # добавляем выходной слой
    model.add(Dense(units=2, activation='softmax'))
    # компилируем модель
    model.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=['accuracy'])
```

Здесь мы создаем сверточную нейронную сеть с двумя сверточными слоями, двумя pooling слоями, одним flatten слоем и двумя полносвязными слоями.

Затем мы компилируем модель, используя оптимизатор 'adam', функцию потерь 'categorical_crossentropy' и метрику 'accuracy'. Мы можем определить другие оптимизаторы или методы для уменьшения потерь, а также добиться более точных результатов путем выбора другой архитектуры нейронной сети.

Шаг 3. Получение исходных данных.

После того, как модель нейронной сети будет обучена, мы можем приступить к тестированию.

Для этого мы загружаем изображения, обрабатываем их и передаем через нашу обученную нейронную сеть. Нейронная сеть производит вычисления и выводит информацию о серии и номере паспорта, которая отображается на экране.

Шаг 4. Доопределение данных.

После получения известных данных, необходимо связаться с базой данных (БД), которая хранит информацию о серии и номере каждого паспорта. В БД необходимо получить доступ к полям, содержащим сведения о серии и номере паспорта. Это осуществляется запросами SQL. Далее необходимо разделить split методом записи серии и номера на отдельные цифры. Потом производится позиционирование недостающих знаков. В итоге работы описанной процедуры будет сформирован список паспортных данных, соответствующий исходным данным. С помощью дополнительных данных: ФИО, место рождения, дата рождения и т.д., выявляется необходимая строка, из которой на выход программы идет серия и номер паспорта.

Таким образом, использование нейронной сети Keras может улучшить процесс доопределения информации о серии и номере паспорта по нечетким исходным данным из-за пролития воды на паспорт.

Несмотря на то, что может потребоваться множество изображений паспортов для обучения нейронной сети, уменьшение времени на доопределение информации может значительно повлиять на процесс идентификации человека.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Фейн Я. Python для профессионалов. – Питер, 2017.

2. Захаров С. Neural Networks and Deep Learning на Python и TensorFlow. – БХВ-Петербург, 2018.

3. Зверев И. Python для анализа данных. – ДМК Пресс, 2018.

4. Библиотека Keras – Русскоязычная документация Keras. – Текст: электронный. – URL: https://ru-keras.com (дата обращения: 15.04.2023).

УДК 004.4

Е.Р. КОМЛЕВА
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ ДЛЯ СТУДЕНЧЕСКОГО КЛУБА НА ОСНОВЕ ПЛАТФОРМЫ «1С: ПРЕДПРИЯТИЕ 8.3»

Рассматривается задача разработки и проектирования информационной системы для студенческого клуба на платформе «1С: Предприятие 8.3».

Студенческий клуб РГРТУ (СК РГРТУ) – самодеятельная студенческая некоммерческая, неполитическая общественная организация при РГРТУ, осуществляющая свою деятельность в соответствии с положением, Уставом РГРТУ и действующим законодательством Российской федерации (РФ), которая занимается организацией и проведением различных мероприятий в образовательном учреждении. Структура включается в себя: председатель СК, заместитель председателя СК, ответственный за партнеров, ответственный за конференс, ответственный за участников, ответственный за техническое обеспечение, ответственный за администраторов сцены.

Для обеспечения координации всех членов СК на протяжении всего процесса подготовки необходим ряд мероприятий. Первый этап включает в себя разделение на задачи определенным лицам. Для решения поставленных целей существуют ряд приложений, которые

помогают качественно подготовить мероприятие: Discord (позволяет обмениваться голосовым, видео и текстовым чатом с организаторами во время мероприятия), Trello (приложение для управления персональными задачами), ВКонтакте (социальная сеть, для создания общих чатов как организаторов, так и участников, легкий обмен файлами).

В результате проведенного анализа, можем прийти к выводу, что полноценных аналогов, позволяющих решать поставленную цель, разрабатываемой системы нет. Каждое приложение, используемое СК, обладает недостаточным функционалом и удобством, а также не может развиваться как полноценная платформа для коммуникации студентов-организаторов. Многие приложения ушли с российского рынка. Поэтому, чтобы не столкнуться с проблемой ухода того или иного продукта удобным решением будет выбор 1С: Преприятия, в котором можно реализовать весь необходимый функционал для информационной системы студенческого клуба РГРТУ.

Достоинства 1С: Предприятия 8.3:

- облачные технологии 1с позволяют на различных операционных системах и разных устройствах;
- экономическая и аналитическая отчетность 1С;
- система прав доступа 1С;
- среда быстрой разработки – позволяет экономить время за счет умной системы, которая сама помогает разработчику создавать качественные прикладные решения в кратчайшие сроки;
- многоплатформенность;
- интеграция происходит практически с любыми внешними программами и оборудованием;
- система взаимодействия позволяет взаимодействовать между собой клиентским приложениям, серверу и пользователям информационной базы.
- отказоустойчивость.

В процессе разработки информационной системы СК РГРТУ были выделены следующие роли и функции:

- программист;
- администратор (председатель СК, заместитель председателя СК): создание мероприятия (наименование, вид, дата, время, место); добавление организаторов; добавление задач ответственному за направление и установка дедлайна; создание чата организаторов; создание конференций;распределение бюджета мероприятия;
- пользователь (ответственный за направление): создание задач для организаторов своего направления; доступ к чату, конференциям и

документам; добавление к чатам и конференциям организаторов своего направления.

Результаты разработки информационной системы на платформе «1С: Предприятие 8.3». На рисунке 1 представлено дерево конфигурации разработанной информационной системы.

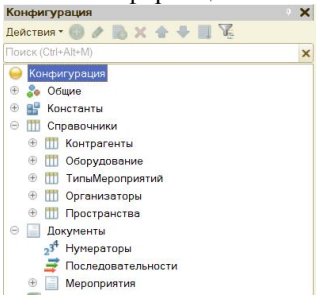


Рисунок 1 – Дерево конфигурации системы

На рисунке 2 показан режим нескольких пользователей для входа в систему Студенческого клуба РГРТУ.

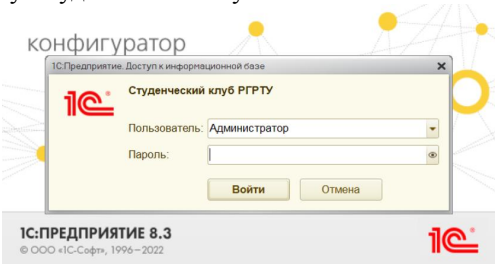


Рисунок 2 – Доступ к информационной базе

Начальная страница разработанной информационной системы представлена на рисунке 3.

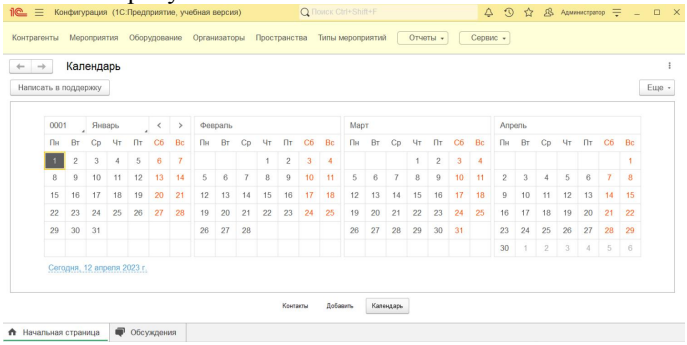


Рисунок 3 – Начальная страница информационной системы

Для того, чтобы создать мероприятие необходимо перейти на вкладку «Мероприятие» и нажать на кнопку «Создать мероприятие». В открывшемся окне необходимо заполнить данные по мероприятию (рисунок 4).

Рисунок 4 – Создание мероприятия Студенческого клуба

В процессе проведенного исследования был проведен анализ предметной области, рассмотрены аналоги и их недостатки и сформирован функционал системы, следующим этапом будет являться разработка системы на платформе «1С: Предприятие 8.3».

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Устав федерального государственного бюджетного образовательного учреждения высшего образования «Рязанский государственный радиотехнический университет». – 2018. – 40 с.

2. Правила внутреннего распорядка обучающихся РГРТУ. – Вып. 1. – 2021. – 11 с.

3. Положение о студенческих клубах и коллективах Рязанского государственного радиотехнического университета. – Вып. 1. – 2010. – 3 с.

4. ФГБОУ ВО «РГРТУ им. В.Ф. Уткина»: официальный сайт. – Текст: электронный. – URL: http://www.rsreu.ru/vuz/vr/tvorcheskie-kollektivy-kluby-kruzhki-sektsii/kluby (дата обращения: 15.04.2023).

УДК 004.021

Д.И. КОНДРАШОВ
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

УСОВЕРШЕНСТВОВАНИЕ АЛГОРИТМА MAPREDUCE И ЕГО ТЕСТИРОВАНИЕ

В статье рассматриваются концептуальные улучшения алгоритма обработки больших данных MapReduce на языке Java, проводится экспериментальное исследование усовершенствованного алгоритма на скорость работы.

В своей статье «Анализ эффективности алгоритма mapreduce при решении задачи подсчёта количества вхождений различных слов в текст» [3] мною был рассмотрен алгоритм MapReduce и его преимущества перед обычным линейным алгоритмом. Теперь я попробую усовершенствовать алгоритм mapreduce, чтобы он работал ещё быстрее и эффективнее. Изначальную постановку задачи подсчёта слов в тексте и результаты исследования можно найти в вышеупомянутой статье.

Подсказки, как можно улучшить алгоритм, приведены в официальной документации Apache [1]. А также в различных статьях [2]. Поскольку эта технология уже существует довольно давно, накопилось немало предложений по её улучшению. Воспользуемся этими знаниями и улучшим алгоритм.

Изначальная реализация алгоритма на языке java имела следующий вид, представленный на рисунке 1 (подробное описание кода приведено в официальной документации [1]).

Первое улучшение – использование шаблонов. В изначальной версии MapReduce считал словами символы, разделённые пробелом. В итоге, например, из текста «Hello World, Bye World! Hello Hadoop, Goodbye to hadoop» алгоритм выдаёт такой результат:

```
Bye 1
Goodbye 1
Hadoop, 1
Hello 2
World! 1
World, 1
hadoop. 1
to 1
```

Благодаря шаблонам мы можем более точно указать, что именно считать словом. Необходимо создать файл шаблона с текстом:

```
 \.
\,
\!
to
```

Таким образом указываем, что необходимо очищать текст от запятых, точек, восклицательных знаков и предлога, например, «to». Теперь алгоритм MapReduce будет более правильно отбирать слова:

```
Bye 1
Goodbye 1
Hadoop 1
Hello 2
World 2
hadoop 1
```

```
public class WordCount {

  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }

  public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values,
                       Context context
                       ) throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Рисунок 1 – Программный код реализации алгоритма MapReduce на языке java в простейшем варианте

В качестве второго улучшения, можно ввести параметр, отвечающий за чувствительность к верхнему или нижнему регистру символов. Его определяет пользователь: если регистр не важен, то весь

текст будет приведён к одному регистру, что ускорит процесс подсчёта слов.

Вместе с двумя вышеупомянутыми улучшениями к коду добавится несколько методов, приведённых на рисунке 2.

```
private boolean caseSensitive;
private final Set<String> patternsToSkip = new HashSet<>();

    @Override
    public void setup(Context context) throws IOException {
            Configuration conf = context.getConfiguration();
            caseSensitive = conf.getBoolean("wordcount.case.sensitive", true);
            if (conf.getBoolean("wordcount.skip.patterns", false)) {
                URI[] patternsURIs = Job.getInstance(conf).getCacheFiles();
                for (URI patternsURI : patternsURIs) {
                    Path patternsPath = new Path(patternsURI.getPath());
                    String patternsFileName = patternsPath.getName();
                    parseSkipFile(patternsFileName);
                }
            }
        }

private void parseSkipFile(String fileName) {
            try {
                BufferedReader fis = new BufferedReader(new FileReader(fileName));
                String pattern;
                while ((pattern = fis.readLine()) != null) {
                    patternsToSkip.add(pattern);
                }
            } catch (IOException ioe) {
                System.err.println("Caught exception while parsing the cached file '"
                        + StringUtils.stringifyException(ioe));
            }
        }
```

Рисунок 2 – Переопределённый метод setup и новый parseSkipFile, добавленные в рамках усовершенствования алгоритма MapReduce

Переопределив метод setup, мы устанавливаем, что нам важна чувствительность регистра:

```
caseSensitive=conf.getBoolean("wordcount.case.sensitive"
, true);
```

А также, в случае наличия файла с шаблоном, выполняем запись во множество patternsToSkip символов из файла с шаблоном, которые нам нужно пропускать при подсчёте слов. Это реализуется с помощью вспомогательного метода parseSkipFile. Теперь метод map будет выглядеть так, как изображено на рисунке 3.

```
@Override
        public void map(Object key, Text value, Context context
        ) throws IOException, InterruptedException {
            String line = (caseSensitive) ?
                    value.toString() : value.toString().toLowerCase();
            for (String pattern : patternsToSkip) {
                line = line.replaceAll(pattern, "");
            }
            StringTokenizer itr = new StringTokenizer(line);
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, one);
                Counter counter = context.getCounter(CountersEnum.class.getName(),
                        CountersEnum.INPUT_WORDS.toString());
                counter.increment(1);
            }
        }
    }
```

Рисунок 3 – Усовершенствованный метод map

В начале этого метода видно, что в случае важности регистра (caseSensitive = true) весь текст приведётся к нижнему регистру (value.toString().toLowerCase()) и останется нетронутым в противном случае. Следующими строчками:

```
for (String pattern : patternsToSkip) {
    line = line.replaceAll(pattern, "");
}
```

выполняется замена всех набранных ранее во множество patternsToSkip символов (которые нужно пропускать при подсчёте слов) в исходном тексте на пустые строки, то есть попросту они убираются из текста. В качестве дополнительного улучшения были использованы счётчики (Counter) – для более быстрого подсчёта слов, а также некоторые другие незначительные особенности.

Итак, после добавленных улучшений, сравним эффективность нового алгоритма со старым, код которого был приведён ранее. Было произведено несколько замеров с подсчётом среднего времени выполнения для нового алгоритма. Для старого алгоритма это время уже приводилось в статье [3] – это примерно 6,87 секунд.

Напомним технические характеристики ЭВМ, на котором проводится исследование: сравнение работы алгоритмов проводилось на базе ноутбука «Dell Vostro 15 3515». Версия JDK: «Oracle OpenJDK version 18.0.1». Интегрированная среда разработки: «IntelliJ IDEA 2022.2.4 (Community Edition)». Процессор: «AMD Ryzen 7 3700U with Radeon Vega Mobile Gfx 2.30 GHz». Размер оперативной памяти: 16,0 ГБ (доступно: 13,9 ГБ). Операционная система – 64-разрядная, «Windows 10 Pro», версия «21H2», сборка «19044.2130».

В качестве входного файла был использован вручную сгенерированный текстовый файл (с расширением «.txt») с набором слов через запятую. Один и тот же файл подавался на вход двух разных программ. Размер файла составляет 172 856 КБ.

Итак, для нового алгоритма среднее время выполнения составило 4,65 секунд, что является незначительным, но все же улучшением уже существующего алгоритма. К тому же, не стоит забывать, что алгоритм стал более гибким – появилась возможность учитывать регистр символов и задавать шаблоны для поиска слов.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. MapReduce Tutorial. – Текст: электронный. – URL: https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html (дата обращения: 23.03.2023).

2. Vlassov V. MapReduce: Limitations, Optimizations and Open Issues / V. Vlassov, V. Kalavri // 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications. – 2013. – С. 1031-1038.

3. Математическое и программное обеспечение вычислительных систем: Межвуз. сб. науч. тр. / Под ред. Г. В. Овечкина. – Рязань: РГРТУ им. В.Ф. Уткина. – 2023. – 124 с.

УДК 004.932

Т.И. КОНДРАШОВА, А.И. ЕФИМОВ
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ИССЛЕДОВАНИЕ МЕТОДОВ ДЕТЕКТИРОВАНИЯ ГРАНИЦ ПЕРЕПАДОВ ЯРКОСТЕЙ НА ИЗОБРАЖЕНИЯХ

В данной работе рассматриваются существующие методы и алгоритмы детектирования границ перепадов яркостей на изображениях, выполнен анализ их преимуществ и недостатков

Методы детектирования границ являются важным инструментом обработки изображений и до сих пор остаются актуальными. Они широко используются в различных областях, включая компьютерное зрение, медицинскую диагностику, автоматическое распознавание лиц, обработку изображений в режиме реального времени, робототехнику,

а также для анализа качества изображений и решения проблем, таких как нечеткость, зернистость, перекрытие объектов и т.д.

Целью данной исследовательской работы является рассмотрение существующих методов и алгоритмов детектирования границ перепадов яркостей на изображениях, рассмотрение их преимуществ и недостатков.

Определение границ объектов на изображениях – это процесс нахождения точек или пикселей, где яркость или цвет изображения резко меняются, что указывает на границы различных объектов на изображении. Эти границы являются важной информацией для дальнейшей обработки изображений, такой как распознавание и классификация объектов, сегментация изображений и другие приложения [1].

Для более точного анализа и обработки изображений вместе с выделением границ изображения используется сегментация.

Сегментация изображения – это процесс разделения цифрового изображения на множество областей или наборов пикселей.

По сути границы могут использоваться в качестве основы для выполнения сегментации изображения. Например, границы могут быть выделены с помощью оператора Собеля, а затем использоваться для сегментации объектов на изображении. Или наоборот, после выполнения сегментации объектов на изображении, границы могут быть выделены для более точного определения контуров объектов.

Для выделения границ используют различные операторы, определяющие изменения градиента уровней серого. Их можно разделить на две основные категории (рисунок 1):

1. Детекторы границ первого рода (оператор, основанный на градиенте);

2. Детекторы границ второго рода (оператор, основанный на Лапласиане).

Детекторы границ первого рода, также называемые операторами, основанными на градиенте, используются для обнаружения границ на изображениях. Они основаны на вычислении градиента яркости в каждой точке изображения и нахождении мест, где градиент достигает максимального значения [2].

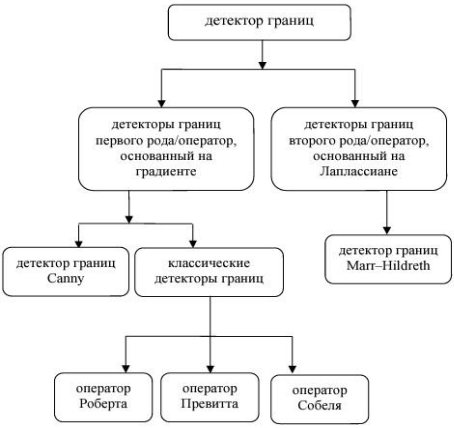


Рисунок 1 – Методы детектирования границ

Одним из наиболее распространенных операторов градиента является операторы:

- Собеля;
- Робертса;
- Превитта;
- детектор границ Кэнни.

Операторы, основанные на градиенте, имеют ряд преимуществ, таких как простота реализации, быстродействие и хорошую чувствительность к границам, но они могут давать ложные срабатывания в шумных изображениях и на текстурах с большим числом деталей (рисунок 2).

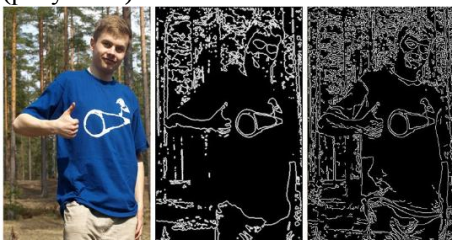


Рисунок 2 – Работа операторов, основанных на градиенте

Детекторы границ второго рода – это операторы, которые используются для обнаружения границ на изображениях и основаны на вычислении второй производной яркости в каждой точке изображения [2]. Они позволяют обнаруживать не только резкие

перепады яркости, но и более плавные границы. Один из наиболее известных детекторов границ второго рода – оператор Лапласа (рисунок 3).

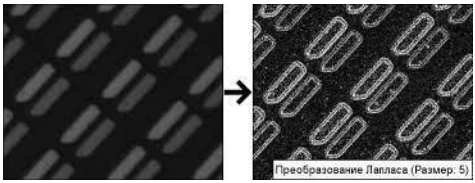


Рисунок 3 – Работа оператора Лапласа

Детекторы границ второго рода обычно дают более точные результаты, чем детекторы границ первого рода, такие как операторы, основанные на градиенте. Однако они также более чувствительны к шумам на изображении и могут приводить к ложным обнаружениям границ.

Рассмотренные детекторы имеют ряд преимуществ, таких как простота реализации, быстродействие и хорошая чувствительность к границам, ускорение процесса обработки данных. Но несмотря на преимущества существующих методов детектирования границ, существуют и недостатки, такие как чувствительность к шумам на изображении, избыточное или недостаточное выделение границ. Чтобы избежать данных недостатков, можно прибегнуть к модификации алгоритмов [3].

Модификация детектирования границ может включать в себя изменение параметров алгоритмов, использование более сложных алгоритмов или комбинацию различных методов для повышения точности и скорости детектирования границ.

Например, одним из методов модификации детектирования границ может быть использование алгоритмов глубокого обучения, таких как сверточные нейронные сети (CNN) [4]. Эти алгоритмы могут обучаться на больших наборах данных изображений и могут обеспечить более точное и быстрое детектирование границ.

Другим методом модификации детектирования границ может быть комбинация нескольких методов детектирования границ. Например, можно использовать метод на основе дифференцирования для выделения границ на изображении и затем использовать оператор Лапласа для уточнения этих границ.

Наконец, модификация детектирования границ может включать в себя оптимизацию параметров алгоритмов для достижения более высокой точности и скорости детектирования границ. Например, можно оптимизировать параметры фильтров, используемых для

детектирования границ, или параметры порогового значения для бинаризации изображения.

В целом, детекторы границ – это полезный инструмент для обработки изображений, но при использовании детекторов границ необходимо учитывать их ограничения и особенности каждого метода, а также возможность наличия ложных срабатываний. В некоторых случаях может потребоваться дополнительная обработка изображения или комбинация нескольких методов, чтобы достичь наилучшего результата.

Таким образом, использование детекторов границ следует рассматривать в контексте конкретной задачи и целей обработки изображений, учитывая их преимущества и недостатки.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Ponomarenko, N. Edge Detection and Filtering of Images Corrupted by Nonstationary Noise Using Robust Statistics / N. Ponomarenko, D. Fevralev, A. Roenko, S. Krivenko, V. Lukin, I. Djurovič // Proceedings of CADSM-2009. – 2009. – Pp. 129-136.

2. Bradley, D. Adaptive Thresholding Using the Integral Image / D. Bradley, G. Roth // Journal of Graphics Tools. – 2007. – №12. – P. 13-21.

3. Maini R., Aggarwal H. Study and comparison of various image edge detection techniques // International Journal of Image Processing. – Vol. 3, N. 1. – 2009. – Pp. 1-11.

4. Науменко А.В., Лукин В.В. Детектирование границ на изображениях с помощью искусственной нейронной сети // Информационные технологии. Авиационно-космическая техника и технология. – 2012. – № 2 (89) – 2012. – С. 101-110.

УДК 004.422.81

И.А. КОРОТКИХ, С.В. АНИКЕЕВ, Д.В. АНИКЕЕВ
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ОБЗОР ПРОБЛЕМАТИКИ УПРАВЛЕНИЯ IT-СЕРВИСАМИ ПРЕДПРИЯТИЯ ЖИЛИЩНО-КОММУНАЛЬНОГО ХОЗЯЙСТВА

Рассматривается основные вопросы, связанные с управлением IT-сервисами жилищно-коммунального хозяйства, даются основные определения, связанные с этой темой, рассматриваются различные программные комплексы, и приводится их сравнительная характеристика.

Проблема правильной обработки платежей ЖКХ имеет важное значение для обеспечения эффективного функционирования жилищно-коммунального хозяйства и удовлетворения потребностей населения в эффективном сервисе.

Неправильная обработка платежей может привести к задержкам в начислении и оплате коммунальных услуг, что в свою очередь может вызвать неудовлетворенность жителей и возникновение проблем взаимоотношений между жителями и органами управления ЖКХ. Кроме того, неправильная обработка платежей может привести к убыткам для бюджета муниципалитета и созданию дополнительных финансовых обязательств для жителей.

Важность правильной обработки платежей ЖКХ также заключается в том, что она является одним из ключевых элементов обеспечения прозрачности и эффективности работы муниципальных органов управления ЖКХ. Точность и своевременность обработки платежей позволяют повысить доверие жителей к органам управления ЖКХ.

Таким образом, правильная обработка платежей ЖКХ является необходимым условием для обеспечения качественных услуг и удовлетворения потребностей населения в комфортном проживании, а также для эффективного функционирования муниципальных органов управления ЖКХ.

В рамках этой проблематики выделим основные понятия.

Биллинговые системы – это системы, вычисляющие стоимость услуг для каждого клиента и хранящие информацию обо всех тарифах, в том числе стоимостных характеристиках. Данные характеристики используются телекоммуникационными операторами для предоставления счетов абонентам [1].

Биллинговые системы по способу выставления счета подразделяются на системы с прямым выставлением счета (Direct Billing System – DBS) и консолидирующие биллинговые системы (Consolidating Billing Systems – CBS)[1].

Взаимодействие между CBS и DBS можно увидеть на рисунке 1:

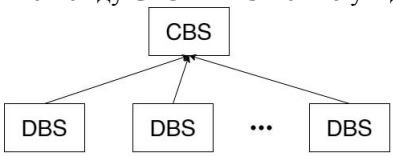


Рисунок 2 – Схема взаимодействия биллинговых систем

В качестве примера CBS можно привести программный комплекс «Абонент +». Он состоит из двух компонентов:

1. Платежная система – является непосредственно CBS. Консолидирует платежи и создает общий счет – единый платежный документ.

2. Расчетная система – это DBS в рамках этой системы взаимодействия.

К программному комплексу могут подключаться следующие виды организаций:

- владельцы обычных биллинговых систем;
- агенты по приему платежей.

Агенты по приему платежей – это точки, через которые происходит транзакция. Например, различные мобильные приложения, банковские приложения и так далее.

В настоящее время проводится работа по упрощению интеграции таких клиентов за счет реализации SDK для Web и мобильных устройств.

SDK (Software Development Kit) используется при создании программного обеспечения. Оно обеспечивает готовые решения для упрощения разработки. Обычно SDK снабжаются всей необходимой документацией и инструментами. Всё это в совокупности позволяет ускорить процесс разработки, уменьшая время, необходимое для выхода на рынок. Кроме того, SDK можно использовать на различных платформах, например, для Web разработки, Desktop разработки, на IOS и Android. Это означает, что у разработчиков имеется возможность создать кроссплатформенную инфраструктуру меньшими усилиями.

В настоящее время в разработке находится SDK «Абонент +». Оно поможет владельцам агентов по приему платежей подключаться к системе по более упрощенной схеме и интегрировать в свои продукты функционал программного комплекса «Абонент +».

У каждой организации, подключенной к программному комплексу, имеется личный кабинет на Портале. Портал – ресурс, который позволяет операторам платежных точек настраивать их, контролировать прохождение платежей и управлять составляющими системы.

В личном кабинете имеется панель управления, в который оператор имеет возможность производить настройку различных параметров и выполнять действия относительно субъектов – точек приема платежей. В частности, есть возможность добавить, удалить услугу, изменить её параметры, задать конфигурацию точки приема,

отправить команду на точку приема, которая выполнится при её запуске.

Программный комплекс в своей основе использует архитектурный стиль RESTful. Он включает в себя следующие принципы:

1. Использование протокола HTTP, а, следовательно, запросов GET, POST, PUT и так далее.
2. Ресурсы доступны по уникальному URI адресу.
3. Все ресурсы имеют только один тип представления.
4. Клиенты должны иметь возможность кешировать данные.

Таким образом, RESTful – это набор правил и ограничений, которые позволяю разработчикам создавать приложения, использующие стандартные протоколы передачи данных. Такие приложения являются гибкими, и их возможно интегрировать с другими приложениями.

Аналогами программного комплекса являются различные программы бухгалтерского устройства. Однако они не учитывают специфику, характерную для сферы услуг ЖКХ, поэтому они не являются прямыми конкурентами. Приведем сравнительный анализ некоторых из них в Таблице 1:

Таблица 1 – Сравнительный анализ

Наименование программного обеспечения	Возможность выставить счет	Возможность консолидировать платежи	Возможность подключения к системе с помощью SDK	Возможность управлять точками приема платежей
Программный комплекс «Абонент+»	Да	Да	Да	Да
«Инфо-Бухгалтер»	Да	Да	Да	Нет
«1С: Бухгалтерия»	Да	Да	Да	Нет
«Турбо-Бухгалтер»	Да	Да	Нет	Нет
«БЭСТ»	Да	Да	Нет	Нет
«Инфософт»	Да	Нет	Нет	Нет

Исходя из приведенного выше анализа можно сделать вывод, что программный комплекс «Абонент+» не имеет прямых конкурентов. В частности, благодаря тому, что имеет возможность обеспечить управляемость точек приема. Но и другие особенности. Такие как возможность интегрирования системы с помощью SDK, возможность

консолидировать платежи так же встречается далеко не у всех аналогов.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. ООО «Абонент+»: Абонент+ Расчетноплатежный комплекс. – Текст: электронный. – URL: https://abonent.plus/ (дата обращения: 01.04.2023).

2. Shen, Y. Research of the Convergent Billing System Based on the Whole Business / Y. Shen, Z.-S. Gu, S. Wang, L. Zhang. – Shanghai: International Conference on Management of eCommerce and e-Government. – 2013. – Pp. 304-307.

УДК 004.422.81

И.А. КОРОТКИХ, С.В. АНИКЕЕВ, Д.В. АНИКЕЕВ
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

РЕАЛИЗАЦИЯ УПРАВЛЕНИЯ IT-СЕРВИСАМИ ПРЕДПРИЯТИЯ ЖИЛИЩНО-КОММУНАЛЬНОГО ХОЗЯЙСТВА

Рассматривается основные подходы для реализации ресурса, позволяющего управлять IT-инфраструктурой предприятия, предоставляющей услуги ЖКХ населению. Описываются используемые при этом технологии, объясняются причины, по которым были использованы именно они.

Перечислим основные признаки правильно реализованной инфраструктуры информационной системы:

1. Масштабируемость: архитектура должна быть способна подстраиваться под изменяющиеся размеры бизнеса.

2. Гибкость: архитектура должна быть гибкой и способной адаптироваться к новым требованиям и технологиям.

3. Эффективность: архитектура должна обеспечивать высокую производительность и быстрый доступ к данным.

4. Безопасность: архитектура должна обеспечивать безопасность данных и защиту от внешних угроз.

5. Простота: архитектура должна быть понятной и простой для понимания и использования.

6. Модульность: архитектура должна состоять из модулей, которые могут быть легко изменены или заменены без влияния на другие части системы.

7. Совместимость: архитектура должна быть совместима с другими системами и технологиями, чтобы обеспечить интеграцию с ними.

8. Надежность: архитектура должна быть надежной и обеспечивать минимальное количество сбоев и ошибок.

9. Удобство использования: архитектура должна быть удобной для использования и обеспечивать простой доступ к функциям и данным.

10. Сопровождаемость: архитектура должна обеспечивать легкость сопровождения и разработки новых функций.

Управление реализуется посредством WEB-приложения. При его создании используется архитектурный паттерн MVC.

MVC позволяет разделить приложения на независимы компоненты, которые можно изменять без опасения повредить другие слои или компоненты. Название этого паттерна является аббревиатурой. Расшифруем её.

M – Model. Представляет собой слой данных. Он отвечает за правильную структуру данных, которые передаются между сервером и клиентом. В частности, модели могут собой классы, в которых атрибуты – это некоторые параметры объекта.

V – View. Это слой, который отвечает за вывод данных на устройство вывода пользователя.

C – Controller. На этом слое происходит необходимая обработка данных. Например, получение данных из базы, их фильтрация, сохранение и редактирование.

Ранее была популярна архитектура, в которой тонкий клиент содержит только представление (рисунок 1).

Несколько позже была создана друга модель, в которой клиент содержит все компоненты, предусмотренные MVC (рисунок 2). Наличие контроллера и моделей позволяют на клиенте производить обработку информации, что может быть полезно в некоторых случаях. Например, задачу валидации данных можно реализовать не только на сервере, но и на клиенте, что повышает не только надежность системы, но и её безопасность, так как через поля, которые не подвергаются валидации могут быть выполнены SQL-инъекции. Это может привести не только к потери данных пользователей, в том числе и чувствительных данных, но и к повреждению самой системы и, возможно даже, к потере контроля над ней.

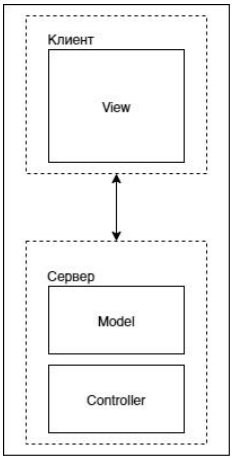


Рисунок 3 – Более ранняя модель

При реализации была использована вторая модель по причине наличия преимуществ, описанных в предыдущем абзаце.

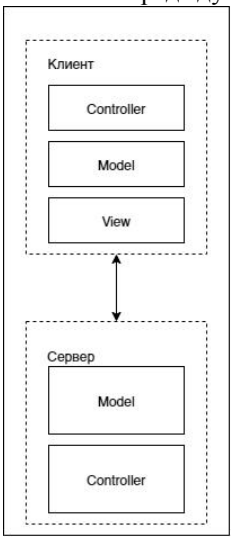


Рисунок 4 – Более новая модель

Интересным также является вопрос аутентификации. Существует 2 основных подхода: аутентификация на основе сессий (рисунок 3) и аутентификация на основе JWT (рисунок 4). Рассмотрим оба подхода.

Алгоритм действий при аутентификации на основе сессий следующий:

1. Пользователь вводит логин и пароль.
2. Приложение на основе введенных данных генерирует токен.
3. Клиент сохраняет токен в cookie.
4. При каждом запросе клиент присылает токен на сервер.
5. При выходе из приложения токен удаляется.

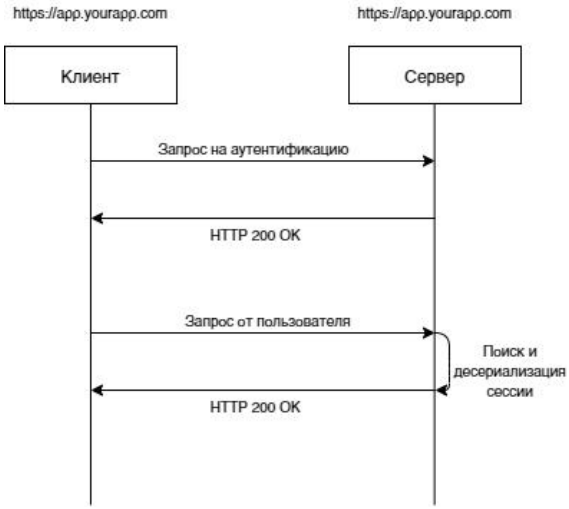


Рисунок 5 – Подход на основе сессий

Особенностью такой системы является то, что необходимо хранить токен на сервере, что приводит к избыточному потреблению памяти. Кроме того, у этой модели есть проблемы с масштабируемостью.

Алгоритм действий при аутентификации на основе JWT-токенов следующий:

1. Пользователь вводит логин и пароль.
2. Приложение на основе введенных данных генерирует JWT токен, хранящий метаданные. Сам токен не хранится на сервере.
3. Клиент сохраняет токен в cookie.
4. При каждом запросе клиент присылает токен на сервер в качестве заголовка или параметра запроса.
5. При выходе из приложения токен удаляется, при этом действий на сервере никаких не происходит, так как токена нет на сервере.

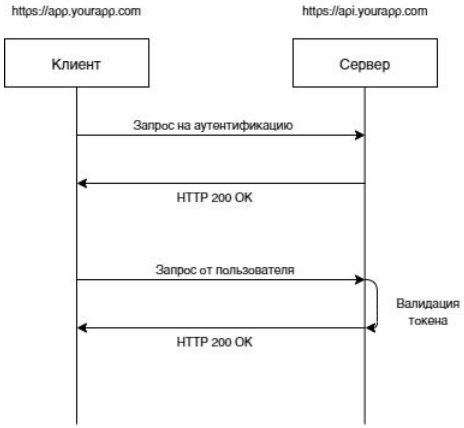


Рисунок 6 – Подход на основе JWT-токенов

Данный подход гораздо лучше подвергается масштабированию и экономит память, поскольку при его использовании не требуется хранить данные на сервере. В разрабатываемом приложении используется именно он.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Introduction to JSON Web Tokens. – Текст: электронный. – URL: https://jwt.io/introduction (дата обращения: 01.04.2023).

2. Pawson R. Naked Objects. – Trinity College, Dublin, Ireland. – 2004. – 397 p.

УДК 004.023

В.П. КОРЯЧКО, А.Н. САПРЫКИН, А.О. САПРЫКИНА
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ОСНОВНЫЕ ТИПЫ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ

Рассматриваются основные типы генетических алгоритмов, их особенности. Описывается принцип их работы, сильные и слабые стороны. Приводится краткая историческая справка о происхождении генетических алгоритмов.

Годом создания генетического алгоритма считается 1975, когда профессор Мичиганского университета Джон Холланд опубликовал первую работу по данной теме. Предпосылками к его созданию

послужили теория эволюции Ч. Дарвина, а также работы и исследования в области эволюции простых автоматов, предсказывающих символы в цифровых последовательностях. Созданный алгоритм был назван «репродуктивный план Холланда», и сразу же стал применяться в самых различных эволюционных вычислениях. Сформулированные Холландом идеи получили свое развитие в работах К. Де Йонга и Д. Голдберга, которые были его учениками, и в итоге был создан широко известный в наши дни классический генетический алгоритм. Помимо описания самого алгоритма были сформулированы описания задействованных операторов и проведено исследование группы тестовых функций.

Генетический алгоритм представляет собой стохастический эвристический алгоритм, применяющийся при решении задач оптимизации и выполнении моделирования с помощью последовательного подбора и рекомбинации, в основе которого лежат процессы, схожие с биологической эволюцией. Данный алгоритм задействует механизм генетического наследования, представляющий собой аналог характерного для природы естественного отбора, сохраняя, хоть и в упрощенном виде, основную терминологию из сферы биологии и основные понятия линейной алгебры. Генетические алгоритмы включены в область мягких вычислений.

В настоящее время термин «генетический алгоритм» употребляется в отношении целого семейства алгоритмов, используемых для решения многих неоднородных типов задач. В качестве примера рассмотрим следующие алгоритмы:

- канонический генетический алгоритм;
- генитор;
- метод прерывистого поиска;
- гибридный алгоритм;
- CHC;
- генетический алгоритм с нефиксированным размером популяции.

Канонический генетический алгоритм

Был предложен Холландом в его работе и, в итоге, в научном сообществе стал именоваться классическим. По правилам канонического генетического алгоритма популяция формируется из N хромосом с фиксированной разрядностью генов. Далее проводится пропорциональный отбор, формируется пул – промежуточный массив, из которого случайным образом выбираются два следующих родителя. На следующем этапе выполняется одноточечный кроссинговер, в результате которого получают двух потомков. После этого созданные

потомки мутируют с заданной вероятностью и занимают места своих родителей. Процесс замены хромосом продолжается до тех пор, пока не будет достигнут критерий завершения алгоритма.

Генитор

Данная модель генетического алгоритма отличается специфичным методом отбора. Начальный этап работы остается неизменным: инициализируется популяция, особи которой оцениваются. Затем случайным образом выбираются две особи, скрещиваются, в результате чего получается только один потомок, который оценивается и заменяет собой не одного из родителей, а наименее приспособленную особь популяции. На следующем шаге снова случайно выбирается пара особей, и их потомок занимает место родительской особи с самой низкой приспособленность. То есть на каждом шаге в популяции обновляется только одна особь. Процесс обновления популяции продолжается до тех пор, пока значения приспособленности хромосом не станут одинаковыми. Алгоритм предусматривает возможность добавления операции мутации потомка, выбор вида кроссинговера и критерия окончания процесса.

Метод прерывистого равновесия

В основу данного метода положена палеонтологическая теория прерывистого равновесия, описывающая быструю эволюцию за счет изменений земной коры. При использовании данного алгоритма после каждой генерации промежуточного поколения особи популяции случайным образом перемешиваются. Потом выполняется основной генетический алгоритм. Для отбора родительских пар используется панмиксия. При такой системе любые две особи имеют одинаковую вероятность создать родительскую пару. Полученные с использованием кроссинговера потомки и родители с наибольшим значением функции приспособленности случайным образом перемешиваются. В новое поколение попадут только особи со значением функции приспособленности выше средней. Таким образом достигается контроль над размером популяции в зависимости от наличия наилучших особей. Данная модификация позволяет уменьшить неперспективные популяции и, напротив, увеличить популяции, содержащие наилучшие индивиды.

Гибридный алгоритм

В основу данного типа алгоритмов положена идея сочетания генетического алгоритма с другим классическим методом поиска, который может применяться для решения поставленной задачи. Потомки, полученные в новом поколении, сначала оптимизируются одним определенным методом, а потом переходят в новую популяцию.

Посредством этого каждая особь целевой популяции может достичь близкого ей локального оптимума. Далее выполняются действия генетического алгоритма: выбор родительских пар, кроссинговер и мутация. Данный тип алгоритмов на практике является очень удачным, так как вероятность попадания одной особи в область глобального максимума крайне велика и в итоге будет найдено решение задачи.

Преимущество такого сочетания состоит в том, что генетический алгоритм позволяет найти во всей области поиска хорошие решения, а оптимизационный метод помогает быстро достичь локального максимума. Тем самым совместное использование этих алгоритмов позволяет удачно сочетать преимущества обоих.

CHC

Алгоритм CHC (англ. Cross-population selection, Heterogeneous recombination and Cataclysmic mutation) достаточно быстро сходится, что объясняется отсутствием мутации после кроссинговера, оперированием популяциями небольшого размера, нестандартной процедурой отбора, заключающейся в том, что отбор ведется как между родительскими особями, так и их потомками. Для кроссинговера выбирается случайная пара, однако не допускается, чтобы между ее элементами было малое хеммингово расстояние или малое расстояние между крайними различающимися битами. Для скрещивания используется разновидность однородного кроссинговера HUC (англ. Half Uniform Crossover). HUC предусматривает переход потомку ровно половины генов от каждого из родителей. Новое поколение формируется из N лучших различных особей родителей и потомков, дублирование не допускается.

Размер популяции алгоритма CHC составляет около 50 особей. Такой относительно небольшой размер популяции оправдывает использование однородного кроссинговера и способствует сходимости метода. Для достижения сходимости в CHC применяется Catalysmic mutation: все хромосомы, кроме самой приспособленной, подвергаются сильной мутации.

ГА с нефиксированным размеров популяции

В данном алгоритме каждая особь имеет дополнительный параметр – максимальный возраст – описывающий число поколений, по истечении который особь погибнет. Введение значения возраста позволило исключить оператор отбора в новую популяцию. Возраст каждой особи популяции рассчитывается индивидуально, исходя из значения ее приспособленности. Каждый раз в каждом поколении формируется и дополнительная популяция из уже полученных

потомков. Производится это на этапе воспроизведения. Для данной цели сначала отбираются особи: их донором становится основная популяция, причем показатель приспособленности не учитывается. Полученные особи получают присвоенный возраст, который соответствует значению приспособленности, он будет постоянной величиной на протяжении всего времени существования каждой отдельно взятой особи. После этого из основной популяции исключаются особи с истекшим сроком жизни, и добавляются потомки из полученной промежуточной популяции.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Гладков, Л.А. Генетические алгоритмы: учебник / Л.А. Гладков, В.В. Курейчик, В.М. Курейчик; под ред. В.М. Курейчик. – Москва: Физматлит, 2010. – 317 с.

2. Фогель Л. Искусственный интеллект и эволюционное моделирование: пер. с англ. / Л. Фогель, А. Оуэнс, М. Уолш; пер. Ю.П. Зайченко; ред. перевода А.Г. Ивахненко; предисл. А.Г. Ивахненко. – Москва: Мир, 1969. – 230 с.

3. Основы генетических алгоритмов. – Текст: электронный. – URL: https://www.intuit.ru/studies/courses/14227/1284/lecture/24168?page=12#sect23 (дата обращения: 12.04.2023).

УДК 622.013

М.К. КРЫГИНА
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ИНТЕРПРЕТАЦИЯ СЕЙСМОАКУСТИЧЕСКОЙ ИНФОРМАЦИИ МОРСКОГО ГЕОФИЗИЧЕСКОГО КОМПЛЕКСА

В статье будет рассмотрена интерпретация сейсмоакустической информации как процесс изучения данных, полученных от сейсмоакустических регистраторов, и будут получены соответствующие практические выводы по дальнейшему использованию извлеченных данных.

В наши дни сейсмоакустика на морских акваториях находит большое распространение как способ изучения строения осадочной толщи и кристаллического фундамента земной коры морей и океанов. В общем случае задача сейсмоакустических исследований формулируется как получение данных, при интерпретации которых

обнаруживаются геологические структуры, определяется их строение и важнейшие свойства.

Интерпретация сейсмоакустической информации заключается в визуальном представлении записей (в печатном виде или на ЭВМ) с их дальнейшей нормировкой (выравниванием) и фильтрацией по разным частотным диапазонам на сейсмотрассе. Главная цель сейсмической интерпретации – это выделение таких волн, в которых содержится полезная информация, на фоне волн, содержащих помехи (как правило «полезные» волны являются немногочисленными по сравнению с посторонними шумами). Затем выполнятся сопоставление отфильтрованных сейсмотрасс, то есть их корреляция.

Во время интерпретации сейсмоакустической информации среди всех волн, полученных на сейсмограммах отбираются те, которые отражаются однократно или преломляются. Далее по динамическим и кинематическим характеристикам этих волн получают распределение скорости волн в залегающих породах. В итоге формируется практическое геологическое толкование – привязка к данным разработки месторождений полезных ископаемых, наблюдение за динамикой образования трещин, изменением почвы, прогнозы динамических проявлений горного давления, составление картины геомагнитных условий изучаемой местности. На основе различия в скорости распределения упругих волн выделяют интервалы среди изучаемой толщи. Их отождествляют с различными породами (отличающимися по составу и возрасту), а также на их основе делают выводы об изменениях в характеристиках пород определенного состава или возраста.

В наши дни геотехника прибегает к помощи ЭВМ для фиксирования, обработки и представления информации, которая накапливается путем различных глубинных измерениях, в том числе и в нестандартных внешних условиях, что дает возможность изменить представление о регистрации и обработке данных от геофизических комплексов. Следовательно, интерпретация и обработка сейсмической информации соединятся в единое целое.

Итогом интерпретации сейсмических данных будет восстановленная волновая картина изучаемой местности. Далее необходимо произвести монтаж сейсмотрасс, дающий информацию об устройстве геологического разреза того участка земной коры, на котором производятся исследования.

Интерпретация геологических данных состоит в преобразовании сейсмогеологического разреза в геологический. Чтобы выполнить такое преобразование необходимо определить количественные

характеристики разреза (количество слоев, глубину их залегания, толщину и форму). Также нужно определить величину скорости распространения упругих волн для каждого интервала между отражающими границами, сопоставить полученную информацию с динамикой работ, изменением почвы, прогнозом скачков в давлении, составить картину тектонических условий района исследования

Временные и глубинные разрезы строятся при помощи узкоспециальных программ на ЭВМ. Следовательно, современные способы сейсмоакустики предполагают фиксацию сейсмоакустической информации в особых форматах данных. Обычно такое программное обеспечение содержит операции обработки результатов методами МПВ и МОВ. Поясним их сущность. В интерпретации сейсмоданных из полезных волн в большинстве случаев пригодны отраженные и преломленные, и, следовательно, методы, основанные на их изучении, называют методом отраженных волн (МОВ) и методом преломленных волн (МПВ).

Рассмотрим метод преломленных волн. Обычно удается выполнить аппроксимацию преломляющих границ в разрезе плоскими поверхностями на линейных профилях, используя этот метод. Тогда говорят, что полученные годографы представляют собой множество плоских слоев.

Для такой границы годограф преломленной волны можно выразить формулой:

$$t(x) = \frac{2H_1}{v_1}\cos i + \frac{x}{v_2}\sin(i \pm \varphi),$$

где υ_1 – эффективная скорость в покрывающей толще; H_1 – эффективная глубина под пунктом возбуждения в заданной точке профиля; φ – угол наклона границы; υ_2 – скорость в породах ниже преломляющей границы.

Обработка информации, полученной от сейсмоакустических регистраторов, методом преломленных волн включает:

1) считывание и визуальное представление сейсмограмм;

2) внесение корректировок в полученные трассы;

3) корреляцию волн;

4) создание и внесение изменений в годографы;

5) измерение сейсмоскоростей границ преломления.

МОВ предполагает проведение исследования способом многократных перекрытий, иначе говоря, методом общей глубинной точки (МОГТ или МОВ-ОГТ).

Граф МОВ-ОГТ включает следующие пункты:

1) ввод информации и присвоение ей геометрического смысла;

2) сортировка трасс по общим точкам (ОТВ) или пунктам (ОПВ);

3) повторение пункта 2 по ОГТ, но с дополнительными исходными условиями (кинематические поправки, пространственно-частотный отбор) и нахождение суммы полученных трасс.

При разработке мест залегания нефти и газа особое значение придается МОВ-ОГТ. Этот метод максимально важен на тех местах, где работы обойдутся дорого, но в то же время качество сейсмоинформации высоко.

Сейсморазведка распространена также в рудной и угольной геологии. При использовании МПВ и МОВ получается фиксировать волны при тектонических сбоях.

Чтобы начать работать с накопляющимися данными, необходимо получить пороговый объем, достаточный для запуска анализа сейсмоакустических данных. Такую обработку возможно проводить на месте проведения исследования данных, и на объекте (фирме, заводе, предприятии). Ввод первичных данных в вычислительную систему (после их предварительного преобразования в устройстве сбора к виду, удобному для последующей обработки) осуществляется либо в режиме непосредственной передачи через соответствующее устройство связи, либо в режиме считывания с промежуточного физического носителя информации. После этого информацию загружают на ЭВМ и обрабатывают, используя специализированное ПО.

Рассмотрим процесс интерпретации сейсмоакустической информации методом отраженных волн. МОВ-ОГТ непростой и многостадийный метод, итогом которого является замена всех сейсмотрасс на сейсмограмме ОГТ одной новой. На ней выделяются однократные волны по сравнению с многократными и другими волнами-помехами. Обработка сейсмоданных МОВ не является автономной, а должна быть под контролем специалиста-геофизика, задающего на ЭВМ параметры работы.

После обработки сейсмоакустических данных на ЭВМ методом отраженных волн получается сейсмический временной разрез (по аналогии с геологическим). На нем отображаются главные отражающие границы, их строение, уровень контраста упругих свойств слоев, находящихся в контакте. На графике по оси абсцисс откладывают интервал от начала профиля, для которого нужно отобразить сейсморазрез. По оси ординат на шкале времени отмечают время распространения отраженных волн от земли до границ отражения и наоборот.

В то же время временной и глубинные разрезы не совпадают. На последнем интервал до границ регистрируют по оси ординат между точкой наблюдения и границей. В заключение обработки временных разрезов выполняется миграция – это трудозатратная математическая обработка, по итогу которой получается глубинный сейсмический разрез. На нем отраженные волны с собственными динамическими свойствами помещаются на реальные глубины и в реальных точках отражения границ.

Итогом комплексной обработки сейсмоакустической информации методом ОГТ являются данные:

1. Структура разрезов на местности, на которых отмечено одна или несколько опорных отражающих границ.

2. Выявленные значения эффективных скоростей в точках исследования профиля.

3. Величины интервальных скоростей для всех интервалов (по отдельности) между границами отражения.

4. Коэффициенты поглощения.

5. Амплитуды отраженных волн или коэффициенты отражения.

На основе исследования рассмотренных выше параметров и сравнение их со свойствами реальных скважин формируется база прогнозирования геологического разреза (ПГР).

Метод отраженных волн чаще используют для исследования строения разрезов осадочных толщ. Этот метод является ведущим в поиске нефти. Второй метод чаще используют на исследованиях под водой на морях и океанах, описании чтроения кристаллического фундамента, поиска залеганий. При инженерно-гидрогеологических исследованиях большую популярность нашел МПВ.

Таблица 1 – Сравнение МОВ и МВП

Признак	МОВ	МВП
Условие, по которому возникает волна	$\sigma_n V_n \neq \sigma_{n+1} V_{n+1}$	$V_{n+1} > V_n$
Уравнение годографа для двухслойной среды (знак "+" по падению, "-" по восстанию пласта)	$t = \frac{1}{V_1} \cdot \cdot \sqrt{x^2 + 4H^2 \pm 4H \sin\varphi}$	$t = \frac{1}{V_1} \cdot \cdot \left[x \sin(i \pm \varphi) + 2H \cos i\right]$
График	гипербола	прямая линия

годографа		
Вариант исследования	сейсмические профилирования и зондирования	сейсмические профилирования и зондирования
Место регистрации волн	вблизи пункта взрыва	вдалеке от пункта взрыва
Частотный спектр	повышенные частоты	пониженные частоты
Данные, получаемые после интерпретации	H, φ, $V_{эф}$	H, φ, $V_г$, менее точно $V_{эф}$
Способы измерения скорости распространения упругих волн	определение $V_{эф}$ в покрывающей толще способом постоянной разности и др.	определение скорости скользящей вдоль границы волны $V_г$ в подстилающем слое по разностному годографу и др.
Способы составления разведываемой границы	построение отражающей границы способами t_0, засечек, эллипсов и др.	построение преломляющей границы способом t_0 и др.

В заключение, хотелось бы еще раз отметить, что интерпретация сейсмических данных заключается в создании по ним, с учетом всей имеющейся априорной информации, геологической модели среды, максимально правдоподобно согласующейся с результатами обработки.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Горяинов Н.Н. Применение сейсмоакустических методов в гидрогеологии и инженерной геологии. – Недра, Москва, 1992. – 264 с.

2. Курьянов Ю.А., Кухаренко Ю.А., Рок В.Е. Сейсмоакустика пористых и трещиноватых геологических сред. Том 1: Теоретические модели в сейсмоакустике поротрещиноватых упругих сред. – ВНИИгеосистем, Москва, 2002. – 202 с.

3. Кирилов А.С., Закревский К.Е. Практикум по сейсмической интерпретации в PETREL. – М.: Издательство маи-принт, 2014. – 288 с.

УДК 004.932

Т.Н. КРЮЧКОВА, А.И. ЕФИМОВ
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ИССЛЕДОВАНИЕ АЛГОРИТМА ОПРЕДЕЛЕНИЯ РАССТОЯНИЯ ДО ОБЪЕКТА С ПОМОЩЬЮ ИЗОБРАЖЕНИЙ СТЕРЕОПАРЫ

В данной статье рассмотрен алгоритм определения расстояния до объекта с помощью изображений стереопары, полученных со стереокамеры.

Определение дальности до объектов с помощью стереопары актуально в настоящее время, так как определение расстояния до объекта по изображениям с двух камер (стереопары) является одной из ключевых задач систем компьютерного зрения.

Для решения данной задачи необходимо изображение стереопары, полученной со стереокамеры (или двух камер – важно, чтобы функциональные характеристики камер были одинаковыми).

Теоретическое выполнение задания состоит в решении математической задачи, представленной ниже. Условие задачи представлено на рисунке 1.

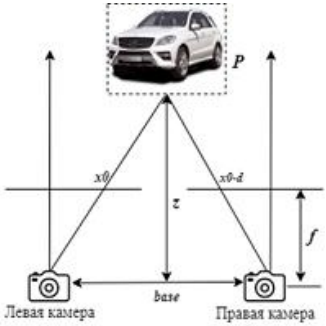


Рисунок 1 – Условие задачи

P – объект, base – расстояние между камерами (м), f – фокусное расстояние камеры стереопары (пкс), x0 – координата по оси OX точки объекта P (с левой камеры), x0-d – координата по оси OX точки объекта P (с правой камеры камеры), d – диспаратность точки OX объекта P (пкс), Z – расстояние до наблюдаемого объекта.

Решение задачи: из рисунка 1 замечаем подобные треугольники. Из этого следует соотношение сторон:

$\frac{base + (x0 - d) - x0}{Z - f} = \frac{base}{Z}$. Приводим подобные: $(x0 - d) - xo = \frac{base * f}{z}$. Отметим, что разница $(x0 - d) - x0$ – значение диспаратности. Из этого получаем отношение: $d = \frac{base * f}{Z}$. Меняем отношение для получения значения Z: $Z = \frac{base * f}{d}$. Задача решена: получили формулу для вычисления расстояния до объекта.

Из решения задачи можно составить программный алгоритм, решающий задачу определения расстояния до объекта с помощью изображений стереопары, полученных со стереокамеры [1].

1. Получение изображений – на камеры поочередно изображения шахматной доски (с разным угловым смещением). Шахматная доска является эталоном для калибровки камеры, т.к. на ней можно наиболее точно определить узлы, по которым в будущем воспроизводиться настройка стереопары.

2. Поиск узлов шаблона на изображениях с обеих камер. Координаты узлов сохраняются для дальнейшего использования.

3. Калибровка каждого из изображений стереопар. Данные в ходе калибровки подаются на вход специальной функции.

4. Калибровка стереопары. После калибровки получаются фокусные расстояния камер и координаты принципиальных точек камер.

5. Устранение дисторсии (нарушения геометрического подобия между объектом и его изображением). После устранения дисторсии все прямые линии объекта становятся прямыми линиями на изображении.

6. Ректификация изображений (для совпадения эпиполярных линий изображения с левой камеры с эпиполярными линиями изображения с правой камеры).

7. Построение карты диспаратностей (карта глубины) для каждого изображения с левой и правой камеры по изображениям, полученным ранее в п.6. Каждый пиксель диспаратностей содержит в себе информацию о том, сколько пикселей на оси Ох находится между соответствующими пикселями изображений между левой и правой камерами.

8. Вычисление расстояния до наблюдаемого объекта. Вычисление расстояния до объекта связано с информацией, полученной с карты глубин.

Схема алгоритма представлена на рисунке 2.

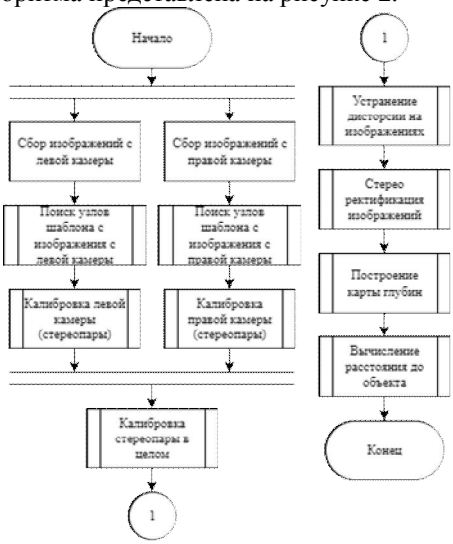


Рисунок 2 – Схема алгоритма определения расстояния до объекта с помощью стереопары

Программный алгоритм реализован на языке Python с помощью библиотек OpenCV, NumPy и других. Код написан в среде разработки IDLE Shell 3.9.7. [0]

Для сбора изображений использовался стенд с скреплёнными друг к другу камерами Logitech C270 с одинаковыми характеристиками. Стенд представлен на рисунке 3.

Рисунок 3 – Стенд со скрепленными камерами

В начале работы необходимо получить пару изображений шахматной доски для калибровки стереопары. Здесь используются функции библиотеки Python – OpenCV для вызова камер по их номерам (cv2.VideoCapture), чтения информации с камер (Cam.read) и

преобразования изображения в формат черно-белого для оптимизации работы. Полученные в ходе работы изображения представлены на рисунке 4.

Рисунок 4 – Изображения с правой и левой камер

Далее происходит поиск узлов шаблона. Используются функции cv2.findChessboardCorners, которая находит положения внутренних узлов шахматной доски; cv2.cornerSubPix для повышения точности координат найденных узлов; cv2.drawChessboardCorners для вывода узлов шаблона на экран монитора. В качестве входных параметров принимаются изображения шахматной доски, размер шахматной доски, узлы с уточненными координатами, логические условия для выполнения операции [0].

Процесс нахождения узлов шаблона изображения шахматной доски представлен на рисунке 5.

Рисунок 5 – Нахождение узлов шаблона

Следующим шагом происходит калибровка камер. Она связана с получением новых, более оптимальных параметров камеры. На вход подаются изображения со стереопары, массив сохраненных узлов шаблона, полученных в предыдущем этапе. Функция cv2.getOptimalNewCameraMatrix принимает новые параметры камер.

Далее происходит калибровка стереокамеры, стереоректификация изображений и построение карты глубин.

Калибровка стереокамеры происходит с помощью функции cv2.stereoCalibrate: оценивается преобразование между двумя камерами, образующими стереопару. Ректификация осуществляется функцией cv2.stereoRectify, на вход подаются встроенные матрицы

левой и правой камер; параметры искажения камер; массив параметров кадров; матрица вращения; матрица перемещения. Функция cv2.initUndistortRectifyMap строит карту глубин (диспаратности) [0]. Построенная карта представлена на рисунке 6.

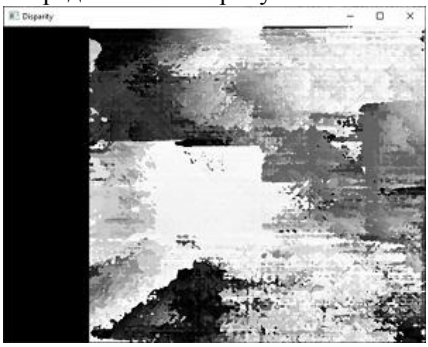


Рисунок 6 – Построенная карта диспаратности

Необходимо скорректировать полученное изображение. Для этого используется функция createRightMatcher. В работе с ней используется WLS фильтр. [0]

Скорректированное изображение карты диспаратности представлено на рисунке 7.

Рисунок 7 – Скорректированная карта глубин

После сбора всей необходимой информации можно решать задачу нахождения расстояния с помощью нее.

Вычисление расстояния до объекта заключается в нахождении связи между величиной диспаратности и реальным расстоянием. В начале работы задаются координаты, с помощью которых будет определяться расстояние до объекта (размер кадра каждый камеры, деленный на 2, чтобы точка поиска была по центру экрана).

Записывается усредненное значения для диспаратности в области точек 3x3. Устанавливаются значения координат точки проекции. Записывается сумма, увеличивающая среднее, как значения искажения, в рамках изменения реальных координат и их проекций. Создается кубическое уравнение для нахождения разницы между величиной искажений и реальным расстоянием [0]:

Distance= -593.97*average**(3) + 1506.8*average**(2) - 1373.1*average + 522.06

Выполнение программы вычисления расстояния до объекта представлены на рисунках 8-10.

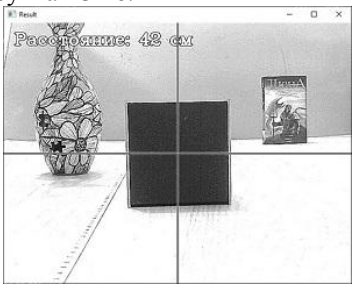


Рисунок 8 – Вычисление расстояния до объекта (42 см)

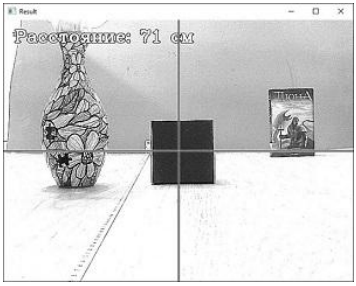


Рисунок 9 – Вычисление расстояния до объекта (71 см)

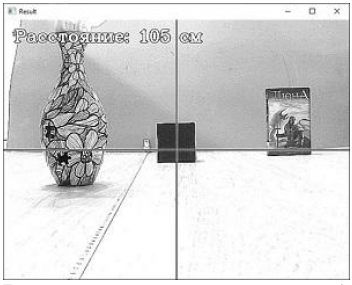


Рисунок 10 – Вычисление расстояния до объекта (105 см)

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Гонсалес Р., Вудс Р. Цифровая обработка изображений. Мир цифровой обработки. Издание 3-е, исправленное и дополненное. – Техносфера, Москва. – 2019. – 1104 с.
2. About Python. – Текст: электронный. – URL: https://www.python.org/about/ (дата обращения: 30.03.2023).
3. Солем Я. Программирование компьютерного зрения на языке Python. – ДМК-Пресс, 2016. – 312 с.
4. Гарсия Г.Б. Обработка изображений с помощью OpenCV / Г.Б. Гарсия, О.Д. Суарес, Х.Л.Э. Аранда, Х.С. Терсеро, И.С. Грасиа, Н.В. Энано. – М.: ДМК Пресс, 2016. – 210 с.
5. Aslam A., Ansari S. Depth-Map Generation using Pixel Matching in Stereoscopic Pair of Images. – Текст: электронный. – URL: https://arxiv.org/abs/1902.03471v3 (дата обращения: 15.04.2023).
6. Fradi H., Dugelay J.L. Improved depth map estimation in StereoVision. – Текст: электронный. – URL: https://www.researchgate.net /publication/253414722_Improved_depth_map_estimation_in_Stereo_Visio n (дата обращения: 15.04.2023).

УДК 004.627

Н.А. КУЗНЕЦОВ, С.В. СКВОРЦОВ
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ПРИМЕНЕНИЕ АЛГОРИТМА ХАФФМАНА ДЛЯ СЖАТИЯ ТЕКСТОВЫХ ДАННЫХ

Рассматриваются особенности применения алгоритма Хаффмана для сжатия текстовых данных с учетом размещения в памяти битовых кодов переменной длины.

Сжатие данных необходимо для уменьшения объема информации, которую нужно передать или сохранить, при этом сохраняя ее качество. Современные технологии все больше требуют обработки больших объемов данных, поэтому сжатие данных играет важную роль в экономии ресурсов и времени.

Существует несколько методов сжатия данных: сжатие без потерь и сжатие с потерями [1]. Сжатие без потерь используется для текстовых документов, архивов, баз данных и других типов данных, где не допустима потеря информации. Сжатие с потерями используется для аудио- и видеофайлов, изображений и других типов данных, где незначительная потеря качества возможна.

Кодирование Хаффмана применяется для сжатия данных без потерь и обеспечивает существенное уменьшение размера сообщения за счет формирования коротких битовых кодов для часто встречающихся символов и более длинных кодовых строк для редко используемых символов.

Алгоритм Хаффмана включает следующие действия [1, 2].

1. Подсчет частоты каждого символа в сообщении.
2. Построение дерева Хаффмана на основе частот символов.
3. Присвоение битовых кодов каждому символу на основе его положения в дереве.
4. Запись закодированного сообщения с использованием полученных битовых кодов.

Например, пусть задан алфавит {A, B, C, D, #} относительные частоты символов которого показаны в таблице 1. Тогда дерево Хаффмана, представленное на рисунке 1, определяет коды символов, приведенные в таблице 2.

Таблица 1 – частоты появления символов

Символ	A	B	C	D	#
Частота	0,35	0,1	0,2	0,2	0,15

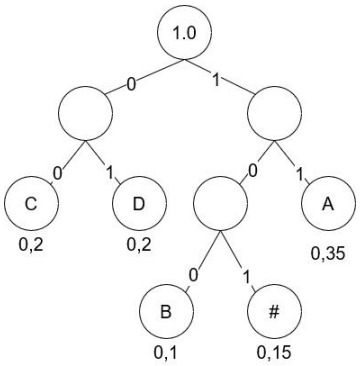


Рисунок 1 – бинарное дерево

Таблица 2 – коды символов

Символ	A	B	C	D	#
Код	11	100	00	01	101

Чтобы получить упакованное закодированное сообщение, каждый символ необходимо заменить на его битовый код и объединить все полученные последовательности в одну строку.

Например, строка DAD кодируется как 011101, а строка BAD#AD как 10011011011101.

Основной проблемой при использовании кодов переменной длины являются определение длины кода текущего символа в закодированном сообщении. Следует отметить, что алгоритм Хаффмана формирует префиксные коды, в которых ни для какого символа алфавита код не является префиксом другого кода. Указанное обстоятельство снимает эту проблему, что позволяет реализовать распаковку достаточно просто.

При распаковке закодированного сообщения выполняется поочередное считывание битов входной строки с одновременным просмотром дерева Хаффмана от корня к листьям. Достижение некоторого листа однозначно определяет символ, на который заменяется прочитанные биты. Затем процесс чтения входной строки возобновляется со следующего бита, а просмотр дерево опять начинается с корня.

Эффективность сжатия для алгоритма Хаффмана оценивается с использованием стандартной меры, которая называется степенью сжатия [1]. Для рассмотренного примера при использовании кодов фиксированной длины требуется не менее чем три бита для каждого символа. Полученные коды переменной длины с учетом заданных частот появления символов дают следующее среднее количество битов для представления одного символа:

$$2*0{,}35+3*0{,}1+2*0{,}2+2*0{,}2+3*0{,}15=2{,}25.$$

Ожидаемая степень сжатия составляет:

$$\frac{(3-2{,}25)}{3}=0{,}25=25\%.$$

На практике степень сжатия для кодирования Хаффмана обычно оказывается в диапазоне от 20% до 80% и зависит от характеристик сжимаемого файла [1].

Основной задачей при программной реализации рассмотренного алгоритма является компактное размещение в памяти очень длинной битовой строки, в виде которой представляется формируемое закодированное сообщение. Для решения этой задачи предложено использовать текстовый файл, заполнение которого производится следующим образом. Полученная битовая строка читается фрагментами по восемь битов и каждый такой фрагмент заменяется символом из стандартной таблицы ASCII, который сохраняется в файле. При этом используется неявное преобразование кода в символ, пример которого показан на рисунке 2. Аналогичный подход

применяется и при распаковке, выполняется неявное преобразование символ в двоичное число.

```
C#

var chars = new[]
{
    'j',
    '\u006A',
    '\x006A',
    (char)106,
};
Console.WriteLine(string.Join(" ", chars));  // output: j j j j
```

Рисунок 2 – пример неявного преобразования

Результаты исследования разработанной программы сжатия текстовых данных представлены в таблице 3.

Таблица 3 – зависимость степени сжатия от длины текста

№ опыта	Число символов в тексте	Степень сжатия	Время работы, с	Начальный размер файла	Размер сжатого файла
1	16	0,3	00,01	16 Байт	11 Байт
2	689	0,46	00,02	1,2 Кб	642 Байт
3	906	0,51	00,02	1,58 Кб	838 Байт
4	3996	0,59	00,10	6,88 Кб	3,67 Кб
5	7845	0,63	00,30	13,5 Кб	7,08 Кб
6	10889	0,66	00,27	18,8 Кб	9,81 Кб

Полученные результаты демонстрируют некоторое несоответствие реальной степени сжатия и ее ожидаемой величины. Объяснением этому могут служить особенности представления текстовых данных в исходном документе. В формате UTF-8 символ может занимать более одного байта памяти, в то время как программа преобразует каждые восемь бит закодированного сообщения в символ, который строго занимает один байт. Таким образом в рамках формата UTF-8 сжатие становится ещё более эффективным.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Кормен Т. Алгоритмы: вводный курс. – М.: ООО «И.Д. Вильямс», 2014. – 208 с.

2. Левитин А. Алгоритмы: введение в разработку и анализ. – М.: Издательский дом «Вильямс», 2006. – 576 с.

УДК 004.415.2.031.43

А.П. ЛЁВКИН
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

РЕАЛИЗАЦИЯ ТЕХНОЛОГИИ СТИЛИЗАЦИИ ИЗОБРАЖЕНИЙ НА ОСНОВЕ ГЕНЕРАТИВНО-СОСТЯЗАТЕЛЬНОЙ СЕТИ

В статье рассматривается реализация технологии стилизации изображений, основанного на нейросетевом подходе GAN и инструментах библиотек обучения нейросетей TensorFlow и Keras

Алгоритмы стилизации изображений на основе нейросетей, также известные как алгоритмы переноса стиля, используют нейронные сети для переноса стиля одного изображения на другое. Эти алгоритмы имеют множество приложений в области компьютерного зрения и графики.

Одним из наиболее популярных алгоритмов переноса стиля является метод, основанный на генеративно-состязательных связях. Этот алгоритм использует предобученную сверточную нейронную сеть для выявления стилевых особенностей одного изображения и применения их к другому изображению.

Принцип работы GAN

GAN – это нейронная сеть, которая обучается генерировать новые данные, похожие на обучающий набор данных [1]. Она состоит из двух частей: генератора и дискриминатора. Генератор создает новые данные, а дискриминатор отличает сгенерированные данные от реальных (рисунок 1).

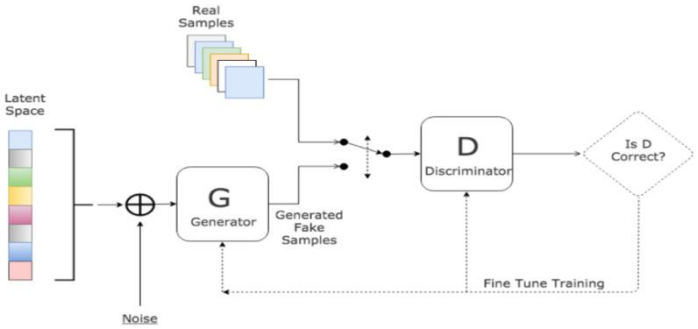


Рисунок 1 – Принцип работы GAN

В основе технологии лежит понятие «состязательной сети», где две нейронные сети работают вместе: генератор и дискриминатор. Генератор создает новые изображения, которые пытаются обмануть дискриминатор, чтобы он принял их за «реальные» изображения. Дискриминатор же обучен различать «реальные» и «сгенерированные» изображения.

В процессе обучения сетей происходит достаточное количество итераций, и результатом является совершенствование генератора, который становится все более точным в создании реалистичных изображений.

Чтобы создать новое стилизованное изображение на основе GAN-сети, обычно используются два изображения: «стиля» и «целевого объекта». Стиль – это изображение, на которое накладывается новый «стиль», то есть изменение визуального восприятия. Целевой объект – это изображение, которое необходимо стилизировать.

Процесс стилизации изображения начинается с передачи стиль-изображения на вход генератора, который анализирует его и генерирует новые параметры, соответствующие стилю. Затем эти параметры передаются на вход дискриминатора, который оценивает, насколько сгенерированное изображение похоже на реальное. Если дискриминатор определяет, что изображение не подходит, генератор получает обратную связь и продолжает генерировать новые параметры. Этот процесс повторяется до тех пор, пока не будет достигнута желаемая точность.

Итоговое изображение, после того как генератор и дискриминатор достигнут согласия, будет сочетать стиль и контент мыслимого объекта. Такой подход к стилизации изображений на основе GAN-сетей дает возможность создавать привлекательные и необычные визуальные эффекты, которые ранее были недоступны с использованием традиционных методов обработки изображений.

Алгоритм стилизации изображений AnimeGan

AnimeGan использует нейронные сети и алгоритмы глубокого обучения для создания и улучшения аниме-изображений [2]. Программа использует технологии, такие как:

1. Автокодировщики (autoencoders) – для сжатия и восстановления изображений, а также для фильтрации шума и артефактов.

2. Генеративно-состязательные сети (GAN) – для создания высококачественных изображений, которые выглядят как настоящие анимационные кадры.

3. Сверточные нейронные сети (CNN) – для обработки изображений и распознавания объектов на изображениях.

4. Рекуррентные нейронные сети (RNN) – для обработки последовательностей данных, таких как последовательности кадров анимации.

5. Технология машинного обучения – для обучения моделей AnimeGan на больших наборах данных.

6. Оптимизация функции потерь – для определения того, как хорошо модель выполняет свои функции, и как можно улучшить результаты.

7. Технологии обработки изображений – для обработки анимационных кадров, изменения их формата, удаления шума и стабилизации видео.

8. Большие данные – для создания анимации с помощью большого набора данных образцов, с целью повышения качества и сохранения фотореалистичности.

AnimeGan предоставляет широкий спектр технических инструментов, которые позволяют производить очень реалистичные изображения аниме. Эти инструменты являются результатом использования передовых технологий в области искусственного интеллекта и компьютерного зрения [3].

Данный алгоритм был протестирован на конкретных изображениях размера 2048*1367. Результаты представлены на рисунках 2-3.

Рисунок 2 – Результат работы алгоритма AnimeGan

Методы улучшения алгоритма

1. Использовать более массивный и разнообразный набор данных для обучения модели.

2. Рассмотреть использование более продвинутых архитектур нейронных сетей, таких как сверточные нейронные сети (CNN),

рекуррентные нейронные сети (RNN), генеративно-состязательные сети (GAN) и др.

3. Изучить итеративный подход к обучению, такой как Transfer Learning, который позволяет использовать предварительно обученные модели для усложнения алгоритма без необходимости обучения модели с самого начала.

4. Уменьшить количество шагов обучения (эпох), чтобы избежать переобучения модели.

5. Обучать модель на более мощном оборудовании, таком как GPU или TPU, чтобы ускорить процесс обучения.

6. Оптимизировать параметры модели, такие как скорость обучения, минимизация потерь и т.д., чтобы наилучшим образом подходить к тренировке модели на тестовом наборе данных.

7. Применять регуляризацию для уменьшения степени сверхобучения модели.

Рисунок 3 – Результат работы алгоритма AnimeGan

В заключение, реализация технологии стилизации изображений на основе генеративно-состязательной сети представляет собой мощный инструмент для создания уникальных искусственных изображений. Она может быть использована в различных областях, от искусства до коммерческих приложений.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Generative adversarial networks. – Текст: электронный. – URL: https://proglib.io/p/generativno-sostyazatelnaya-neyroset-vasha-pervaya-gan -model-na-pytorch-2020-08-11 (дата обращения: 13.04.2022).

2. Генеративные состязательные нейронные сети (GAN-генеративки). – Текст: электронный. – URL: https://www.bizkit.ru/2019/12/04/15391/ (дата обращения: 13.04.2022).

3. Генерация произвольных реалистичных лиц с помощью ИИ. – Текст: электронный. – URL: https://habr.com/ru/post/428221/ (дата обращения: 13.04.2022).

УДК 004.65

А.С. ОЛЬЧЕВ
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ В СЕЛЬСКОМ ХОЗЯЙСТВЕ

Рассматриваются основные направления использования искусственного интеллекта в сельском хозяйстве, существующие решения, его влияние на экологию и перспективы развития.

Искусственный интеллект (ИИ) является одной из наиболее обсуждаемых технологических тенденций в современном мире. Он уже применяется в различных сферах жизни, таких как медицина, финансы, производство, транспорт, сельское хозяйство и т.д.

В медицине ИИ используется для анализа больших объемов медицинских данных, помогая врачам в диагностике и лечении заболеваний. В финансовой сфере ИИ используется для анализа рынков и принятия решений по инвестированию. В производстве ИИ применяется для оптимизации производственных процессов и повышения эффективности работы.

Также ИИ уже используется в автомобильной индустрии для создания автомобилей с автопилотом, в сфере образования для создания индивидуальных программ обучения, а также в сфере развлечений и игр.

В последние годы ИИ стал широко использоваться в сельском хозяйстве. Использование ИИ в сельском хозяйстве может значительно увеличить производительность и эффективность. Например, он может помочь фермерам в принятии решений, таких как выбор оптимального времени для посева, определение оптимального

уровня удобрений и определение оптимального времени для сбора урожая.

Однако, с развитием ИИ возникают новые проблемы и риски, такие как потеря рабочих мест, проблемы конфиденциальности данных и этические вопросы, связанные с созданием автономных систем, способных принимать решения без участия человека.

Примеры использования искусственного интеллекта в сельском хозяйстве

Использование ИИ в сельском хозяйстве может значительно увеличить производительность и эффективность. Ниже приведены несколько примеров.

1. Анализ почвы с использованием искусственного интеллекта может помочь оптимизировать сельскохозяйственное производство, увеличить урожайность и уменьшить негативное воздействие на окружающую среду.

Системы ИИ могут использоваться для анализа данных о почве, таких как ее состав, структура, плотность, влажность и т.д. С помощью алгоритмов машинного обучения, системы могут определять оптимальные условия для выращивания определенных культур и рекомендовать оптимальные методы обработки почвы удобрениями.

Также системы ИИ могут использоваться для мониторинга качества почвы и предотвращения ее загрязнения. С помощью анализа данных о содержании пестицидов, гербицидов и других химических веществ в почве, системы могут предупреждать о возможных проблемах и рекомендовать оптимальные методы очистки.

Однако, для использования систем ИИ для анализа почвы необходимо иметь большой объем данных, что может быть проблемой для малых фермерских хозяйств. Также необходимо учитывать, что системы ИИ не могут заменить опыт и знания опытных сельскохозяйственных работников, а могут использоваться только в качестве инструмента для оптимизации производства.

2. Использование искусственного интеллекта в сельском хозяйстве может значительно улучшить эффективность и точность полива растений.

Одним из примеров такого применения является система полива, основанная на ИИ, которая может автоматически определять оптимальное время и объем воды для полива растений на основе данных о погоде, типе почвы, фазе роста растений и других факторов.

Такие системы могут использовать датчики и сенсоры для сбора данных о влажности почвы, температуре воздуха, влажности и температуре листьев и других параметрах, а затем анализировать эти

данные с помощью алгоритмов ИИ, чтобы определить оптимальное время и объем полива.

Это может помочь уменьшить потребление воды и улучшить качество урожая, так как растения будут получать оптимальное количество влаги в нужное время.

Кроме того, системы полива на основе ИИ могут быть связаны с другими системами управления сельским хозяйством, такими как системы удобрения и защиты растений, чтобы обеспечить более интегрированный и эффективный подход к сельскому хозяйству.

3. Использование искусственного интеллекта может помочь сельским хозяйственным предприятиям быстро и точно определять заболевания сельскохозяйственных культур, что может привести к более эффективному управлению урожаем и уменьшению потерь.

Одним из примеров такого применения является использование нейронных сетей для анализа изображений растений и выявления признаков, указывающих на наличие заболеваний. Например, нейронные сети могут быть обучены распознавать пятна на листьях, деформации стеблей или другие признаки, которые могут указывать на конкретные заболевания.

Кроме того, системы ИИ могут использовать данные о погоде, типе почвы, уровне влажности и других факторах, чтобы более точно определить, какие заболевания могут появиться в конкретной культуре.

4. Определение оптимальных условий хранения: Использование искусственного интеллекта для определения оптимальных условий хранения в сельском хозяйстве может помочь сельскохозяйственным предприятиям снизить потери продукции и увеличить ее срок хранения.

Одним из примеров такого применения является использование алгоритмов машинного обучения для анализа данных о температуре, влажности, освещенности и других факторах, которые могут влиять на качество и срок хранения продукции. Например, ИИ может использоваться для определения оптимальной температуры и влажности для хранения конкретного вида продукции, такой как овощи или фрукты.

Кроме того, системы ИИ могут использовать данные о погоде, времени года, типе почвы и других факторах, чтобы более точно определить, какие условия хранения будут наилучшими для конкретной культуры.

Это может помочь сельскохозяйственным предприятиям принимать более обоснованные решения о том, как хранить свою

продукцию, чтобы она сохраняла свои качественные характеристики наибольшее количество времени.

Развитие ИИ в сельском хозяйстве в мире

Развитие и применение искусственного интеллекта в сельском хозяйстве является одним из ключевых направлений современной аграрной науки и технологии. В мире существует множество проектов, которые используют ИИ для оптимизации производства и повышения урожайности.

Одним из примеров является американская компания Blue River Technology, которая разработала робот-опрыскиватель для уничтожения сорняков. Робот использует компьютерное зрение и машинное обучение, чтобы определить, какие растения являются сорняками, и наносит точечные дозы гербицида только на них.

Компания FarmWise из США разработала робота-сборщика овощей, который использует машинное зрение и глубокое обучение для определения зрелости овощей и сбора их с минимальным повреждением.

В Китае компания XAG разработала систему беспилотных летающих аппаратов для опрыскивания полей. Система использует искусственный интеллект для определения оптимального времени и количества опрыскивания.

В Сингапуре была создана компания Singrow, которая использует искусственный интеллект для управления вертикальными фермами. Система контролирует освещение, температуру, влажность и другие параметры, чтобы обеспечить максимальную урожайность.

В России также ведутся исследования в области применения ИИ в сельском хозяйстве. Например, компания VisionLabs разрабатывает систему компьютерного зрения для определения зрелости ягод на кустах.

В целом, ИИ может значительно повысить эффективность сельского хозяйства и улучшить производство с экономической точки зрения, что позволит удовлетворить растущий спрос на продукты питания в мире и улучшить жизнь миллионов людей. Таким образом, использование искусственного интеллекта в сельском хозяйстве является актуальным и перспективным направлением развития, которое уже находит применение во многих странах мира.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Скворцов Е.А. Перспективы применения технологий искусственного интеллекта в сельском хозяйстве региона // Экономика региона. – 2020. – Т. 16, вып. 2. – С. 563-576

2. Как искусственный интеллект помогает сельскому хозяйству // Про Искусственный Интеллект. – Текст: электронный. – URL: https://dzen.ru/a/YEkyzrEBrkLWMtD2 (дата обращения: 25.03.2023).

3. Потапов А.С. Искусственный интеллект и универсальное мышление / А.С. Потапов – СПБ.: Политехника, 2012. – 711 с.

УДК 004.65

А.С. ОЛЬЧЕВ
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ТЕХНОЛОГИИ КРИПТОВАЛЮТЫ И БЛОКЧЕЙН

Рассматриваются технологии блокчейн и криптовалюты, их использование в России и в мире, влияние криптовалюты на рынок видеокарт и её законность в России.

Блокчейн и криптовалюта – это две технологии, которые стали очень популярными в последние годы.

Блокчейн в настоящее время является одной из наиболее перспективных и быстро развивающихся технологий. Он используется в различных сферах, включая финансы, медицину, правительство, логистику, энергетику и в других.

Криптовалюта в настоящее время продолжает развиваться и набирать популярность. Наиболее известной и широко используемой криптовалютой является Биткоин, но существует множество других криптовалют, таких как Эфириум, Рипл, Лайткоин и другие. Главное её предназначение это проведение платежей и транзакций без участия посредников, таких как банки и платежные системы.

Технология блокчейн

Блокчейн – это распределенная база данных, которая хранит информацию о транзакциях и сделках между участниками сети. Она построена на принципе децентрализации, что означает, что нет единого центрального узла управления, а каждый участник сети имеет доступ к данным и может их изменять. Блокчейн состоит из блоков, каждый из которых содержит информацию о предыдущем, что обеспечивает целостность и безопасность данных и составляет цепь блоков (в переводе на английский: blockchain).

Основные принципы блокчейна:

1. Децентрализация – нет центрального узла управления.

2. Прозрачность – все транзакции видны всем участникам сети.

3. Надежность – блокчейн защищен криптографическими методами.

4. Безопасность – блокчейн защищен от взлома и изменения данных.

5. Низкие комиссии – блокчейн позволяет снизить комиссии за транзакции.

Применение блокчейна:

1. Криптовалюты – блокчейн используется для создания и обмена криптовалют.

2. Учет и хранение данных – блокчейн может использоваться для хранения и учета данных в различных сферах.

3. Смарт-контракты – блокчейн может использоваться для создания и выполнения смарт-контрактов.

4. Идентификация – блокчейн может использоваться для создания систем идентификации.

5. Логистика – блокчейн может использоваться для отслеживания грузов и оптимизации логистических процессов.

Блокчейн является одной из самых перспективных технологий, которая может изменить многие области жизни и бизнеса.

Криптовалюта

Криптовалюта – это цифровая валюта, которая использует криптографию для обеспечения безопасности и анонимности транзакций. Криптовалюты не зависят от центральных банков или правительств, и их цена определяется спросом и предложением на рынке.

Самой популярной криптовалютой является Биткоин, который был создан в 2009 году. С тех пор появилось множество других криптовалют, таких как Эфириум, Лайткоин, Рипл и др.

Криптовалюты могут быть использованы для проведения онлайн-транзакций без необходимости доверять третьей стороне, такой как банк. Они также могут использоваться для инвестирования.

Одним из основных преимуществ криптовалют является их децентрализованность и анонимность. Однако, они также могут быть использованы для незаконных действий, таких как отмывание денег или финансирование терроризма.

Криптовалюты могут быть приобретены на специализированных биржах, таких как Binance, Coinbase, Kraken и др. Однако, перед покупкой криптовалюты необходимо хорошо изучить рынок и принять осознанное решение.

В целом, криптовалюты и блокчейн технология представляют собой новые возможности для экономики и финансовой системы.

Однако, они также требуют дополнительного регулирования и обеспечения безопасности пользователей.

История развития криптовалюты

История криптовалют началась в 2009 году, когда была создана первая и самая известная криптовалюта – Bitcoin. Ее создателем является Сатоши Накамото, который оставил анонимную записку, описывающую принципы функционирования криптовалюты и технологию блокчейн.

С тех пор было создано множество других криптовалют, каждая из которых имеет свои особенности и принципы работы. Например, Ethereum была создана в 2015 году и позволяет создавать децентрализованные приложения на базе блокчейна. Ripple была создана в 2012 году и используется для проведения международных платежей.

Со временем криптовалюты стали все более популярными, привлекая внимание многих инвесторов и трейдеров. В 2017 году произошел бум криптовалют, когда цены на них резко выросли, привлекая множество новых инвесторов и спекулянтов.

Сегодня криптовалюты продолжают развиваться и привлекать все большее внимание со стороны правительств. Некоторые страны уже начали регулировать криптовалюты, в то время как другие продолжают относиться к ним с недоверием.

Криптовалюта в россии

Криптовалюта в России официально не признана средством платежа, но ее использование не запрещено. В 2019 году в России был принят закон «О цифровых финансовых активах», который устанавливает правила обращения с криптовалютами и регулирует деятельность криптобирж.

Согласно закону, криптовалюты могут быть использованы для инвестиций, покупки товаров и услуг в интернете, но не могут быть использованы как средство платежа в России.

Также закон устанавливает требования к криптобиржам, включая обязательную регистрацию и соответствие определенным стандартам безопасности. Кроме того, закон предусматривает возможность создания в России собственной криптовалюты Центральным банком.

Несмотря на это, в России все еще существуют ограничения на использование криптовалют, и многие банки и финансовые учреждения не принимают их как средство платежа. Также в последнее время были предприняты попытки запретить майнинг криптовалют и ограничить доступ к криптобиржам.

Дефицит на видеокарты из-за криптовалюты

Дефицит на видеокарты из-за криптовалюты возник в период активного роста курса биткоина и других криптовалют. Майнеры, занимающиеся добычей криптовалют, стали активно приобретать видеокарты для ускорения процесса майнинга.

Это привело к тому, что на рынке начался дефицит видеокарт, и цены на них резко выросли. Производители видеокарт также не успевали удовлетворить спрос, и многие магазины ограничивали продажу видеокарт на одного покупателя.

Дефицит на видеокарты из-за криптовалюты был временным явлением и связан с конкретной ситуацией на рынке криптовалют. Когда курс биткоина начал снижаться, майнеры перестали активно покупать видеокарты, и на рынке начался обвал цен.

Сегодня на рынке видеокарт можно наблюдать стабилизацию цен и наличие достаточного количества товара. Однако, спрос на видеокарты продолжает оставаться высоким, так как они используются не только для майнинга, но и для игр, графического дизайна и других целей.

Использование криптовалюты в мире

Криптовалюта используется в мире в различных сферах деятельности. Ниже приведены некоторые из них:

1. Инвестирование: многие люди используют криптовалюту для инвестирования в надежные криптовалютные проекты, такие как биткоин, эфириум и другие.

2. Онлайн-платежи: криптовалюта может быть использована для онлайн-платежей, таких как оплата за товары и услуги в интернет-магазинах.

3. Международные переводы: криптовалюта может быть использована для международных переводов денег без необходимости прохождения через банковские системы.

4. Майнинг: майнеры используют криптовалюту для добычи новых блоков в блокчейне и получения вознаграждения в виде криптовалюты.

5. Игры: некоторые игры используют криптовалюту в качестве внутриигровой валюты, которую можно использовать для покупки игровых предметов.

6. Благотворительность: криптовалюта может быть использована для благотворительных целей, таких как пожертвования на различные благотворительные организации.

7. Защита личных данных: криптовалюта может быть использована для защиты личных данных пользователей, так как транзакции в блокчейне не могут быть изменены или подделаны.

Это лишь некоторые из примеров использования криптовалюты в мире. Криптовалюта все больше проникает в различные сферы деятельности и может быть использована для решения различных задач.

Несмотря на все преимущества, блокчейн также имеет свои недостатки, такие как низкая производительность и высокая стоимость майнинга. Однако благодаря постоянному развитию технологии и появлению новых приложений, блокчейн продолжает наращивать свою популярность и использоваться в различных сферах жизни.

Что касается криптовалюты, то она становится все более популярной и широко используемой в различных сферах жизни. Ее будущее остается неопределенным, но существует потенциал для дальнейшего роста и развития.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Swan M. Blockchain. Blueprint for a New Economy. // O'Reilly Media Inc. – 2015. – 123 p.

2. Генкин А. Блокчейн: Как это работает и что ждет нас завтра / А. Генкин, А. Михеев. – М.: Альпина Паблишер, 2018. – 680 с.

3. Криптовалюта для новичков. Как начать пользоваться Биткоином. – Текст: электронный. – URL: https://habr.com/ru/post/352974/ (дата обращения: 15.03.2023).

УДК 004.7

Д.А. ПЕРЕПЕЛКИН, К.В. АНИСИМОВ
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ОСОБЕННОСТИ РЕАЛИЗАЦИИ АЛГОРИТМА ПАРНЫХ ПЕРЕХОДОВ В ПРОГРАММНО-КОНФИГУРИРУЕМОЙ СЕТИ НА ЯЗЫКЕ PYTHON

Описываются основные моменты реализации алгоритма парных переходов в программно-конфигурируемой сети на языке Python.

При решении задачи маршрутизации в программно-конфигурируемых сетях (ПКС) часто используют алгоритмы Дейкстры [1], Беллмана – Форда [2] и Флойда – Уоршелла [3]. Однако данные

алгоритмы имеют существенный недостаток: при изменении метрики канала связи необходимо затрачивать время на построение новых оптимальных маршрутов. Это время в значительной степени зависит от количества коммутаторов в сети. Данного недостатка лишен алгоритм парных переходов [4, 5]. Его особенность состоит в том, что он собирает информацию о сети и вычисляет резервные каналы связи до того, как в сети произошли изменения. Следовательно, при изменении метрики канала связи алгоритм сразу переключает трафик на резервный канал, а затем производит перерасчет той части сети, в которой произошли изменения.

На рисунке 1 изображена экспериментальная топология ПКС.

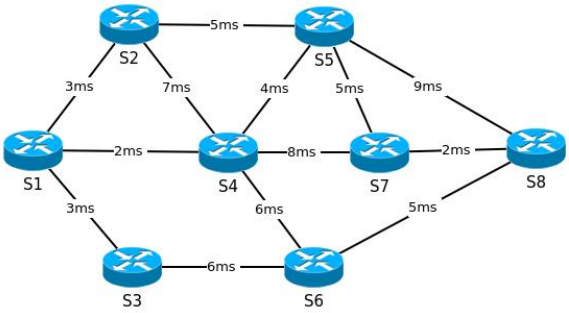


Рисунок 1 – Экспериментальная топология ПКС

При реализации алгоритма парных переходов на языке Python сеть представляется в виде двумерного списка *matrix*, являющегося матрицей смежности сети. Также необходимо создать переменную *src*, которая будет хранить номер начального узла дерева оптимальных маршрутов. Стоит обратить внимание на то, что в Python нумерация списков начинается с нуля. Так, коммутатору S1 на рис. 1 будет соответствовать номер 0. *Matrix* и *src* являются входными данными для работы алгоритма Дейксты, с помощью которого вычисляется дерево оптимальных маршрутов *min_tree*. Само дерево представляется в виде списка двухэлементных кортежей. Каждый кортеж является каналом связи. Для вычислений каналы связи, входящие в *min_tree*, удобно считать ориентированными. Тогда первый элемент – номер начального узла канала, а второй – номер конечного узла канала. Один канал связи должен входить в *min_tree* только один раз. Например, если в *min_tree* уже имеется канал (0, 1), то в нем не должно быть канала (1, 0).

В таблице 1 приведены описания переменных, необходимых для работы алгоритма.

Таблица 1. Описание переменных

Название	**Тип**	**Описание**
switches	Множество	Содержит номера всех коммутаторов сети
routes	Словарь	Ключи – номера коммутаторов. Значения – маршруты от *src* до этих коммутаторов. Маршрут представляется списком номеров коммутаторов, через которые проходит этот маршрут. Значение, соответствующее ключу *src*, является пустым списком.
routes_lengths	Словарь	Ключи – номера коммутаторов. Значения – суммы метрик каналов соответствующих маршрутов. Значение, соответствующее ключу *src*, равняется нулю.
links	Список	Содержит кортежи – каналы связи, которые не вошли в *min_tree*. Один канал связи должен входить в *links* только один раз. Например, если в *links* уже имеется канал (1, 2), то в нем не должно быть канала (2, 1).
adjacent	Словарь	Ключи – номера коммутаторов. Значения – списки коммутаторов, смежных с коммутаторами-ключами.
parent	Словарь	Ключи – номера коммутаторов. Значения – номера коммутаторов, которые являются родительскими к коммутаторам-ключам в дереве оптимальных маршрутов. Значение, соответствующее ключу *src*, равняется None
descendants	Словарь	Ключи – номера коммутаторов. Значения – списки коммутаторов, которые являются потомками коммутаторов-ключей в дереве оптимальных маршрутов
tag	Словарь	Ключи – номера коммутаторов. Значения – значения меток коммутаторов-ключей. Метка показывает сколько каналов связи содержится в минимальном маршруте,

		соединяющим *src* и коммутатор-ключ. Значение метки для *src* равно нулю.
lists	Список	Содержит номера коммутаторов, которые являются листами дерева оптимальных маршрутов *min_tree*. Коммутатор попадает в *lists*, если он не имеет коммутаторов-потомков.

Для топологии на рисунке 1 значения вышеописанных переменных представлены в таблице 2.

Таблица 2. Значения переменных для экспериментальной топологии

Название	**Значение**
src	0
min_tree	[(0, 1), (0, 2), (0, 3), (3, 4), (3, 5), (3, 6), (6, 7)]
switches	{0, 1, 2, 3, 4, 5, 6, 7}
routes	{0: [], 1: [0, 1], 2: [0, 2], 3: [0, 3], 4: [0, 3, 4], 5: [0, 3, 5], 6: [0, 3, 6], 7: [0, 3, 6, 7]}
routes_lengths	[0, 3, 3, 2, 6, 8, 10, 12]
links	[(1, 3), (1, 4), (2, 5), (4, 6), (4, 7), (5, 7)]
adjacent	{0: [1, 2, 3], 1: [0, 3, 4], 2: [0, 5], 3: [0, 1, 4, 5, 6], 4: [1, 3, 6, 7], 5: [2, 3, 7], 6: [3, 4, 7], 7: [4, 5, 6]}
parent	{0: None, 1: 0, 2: 0, 3: 0, 4: 3, 5: 3, 6: 3, 7: 6}
descendants	{0: [1, 2, 3], 1: [], 2: [], 3: [4, 5, 6], 4: [], 5: [], 6: [7], 7: []}
tag	{0: 0, 1: 1, 2: 1, 3: 1, 4: 2, 5: 2, 6: 2, 7: 3}
lists	[1, 2, 4, 5, 7]

Дерево оптимальных маршрутов для экспериментальной топологии изображено на рисунке 2.

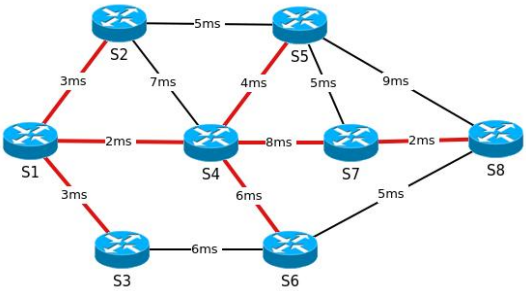
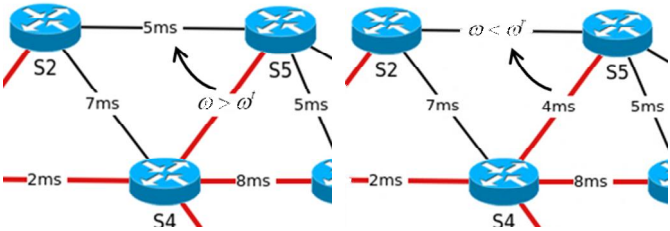


Рисунок 2 – Дерево оптимальных маршрутов экспериментальной топологии

Парный переход может произойти в двух случаях:

1) если значение метрики канала связи, находящегося а дереве оптимальных маршрутов, превысит некоторый порог ω^t, называемый точкой выхода из дерева (рисунок 3 а);

2) если значение метрики канала связи, не входящего в дерево оптимальных маршрутов, станет меньше некоторый порога ω^r, называемого точкой входа в дерево (рисунок 3 б).



а) б)

Рисунок 3 – Виды парных переходов парных переходов

В случае увеличения метрики канала, который не входит в дерево оптимальных маршрутов или в случае уменьшения метрики канала, входящего в него, парного перехода не происходит так как маршруты остаются оптимальными.

Для работы алгоритма также необходимо создать два пустых словаря *transitions*1 и *transitions*2. В словарь *transitions*1 будет заноситься информация о парных переходах, в случае увеличения метрики канала связи, находящегося в дереве оптимальных маршрутов, а в *transitions*2 – в случае уменьшения метрики канала связи, не находящегося в дереве оптимальных маршрутов.

Словарь *transitions*1 в качестве ключей будет иметь кортежи-каналы, входящие в дерево оптимальных маршрутов. Значениями также будут являться словари, ключи которых – некоторые значения метрик, а значения – номера коммутаторов, на которые может произойти переключение канала. Одна пара ключ – значение словаря *transitions*1 будет иметь следующий вид:

$$(v_1^i, v_2^i) : \{\omega_1^t : v_1, \omega_2^t : v_2, \cdots, \omega_j^t : v_j, \omega_M^t : v_M\},$$

где v_1^i – номер первого коммутатора *i*-го канала связи, а v_2^t – номер второго коммутатора *t*-го канала связи. *i*-й канал входит в дерево оптимальных маршрутов. v_j – это номер некоторого коммутатора, смежного с v_2^i, на который заменится коммутатор v_1^i, в случае, если значение метрики *i*-го канала превысит ω_j^t. ω_j^t можно назвать

потенциальной точкой выхода из дерева, а пару $\omega_j^t : v_j$ – потенциальным парным переходом. Очевидно, что значение точки выхода из дерева для *i*-го канала связи будет определяться как:

$$\omega_i^t = \min\{\omega_j^t\},$$

где $j = \overline{1, M}$, M – это число потенциальных парных переходов, которые может совершить *i*-й канал связи.

Таких пар в словаре *transitions*1 будет столько, сколько каналов в дереве оптимальных маршрутов, т. е. $i = \overline{1, N-1}$, N – количество коммутаторов в сети.

Словарь *transitions*2 в качестве ключей также будет иметь кортежи-каналы, но не входящие в дерево оптимальных маршрутов. Значениями также будут являться словари, но только с одной парой ключ – значение. Одна пара ключ – значение словаря *transitions*2 будет иметь следующий вид:

$$(v_1^i, v_2^i) : \{\omega^r : v\},$$

где v_1^i – номер первого коммутатора *i*-го канала связи, а v_2^i – номер второго коммутатора *i*-го канала связи. *i*-й канал не входит в дерево оптимальных маршрутов. v – это номер некоторого коммутатора, смежного с v_2^i, на который заменится коммутатор v_1^i, в случае, если значение метрики *i*-го канала станет меньше ω^r.

Таких пар в словаре *transitions*2 будет столько, сколько имеется потенциальных парных переходов для всех кортежей-каналов в *transitions*1.

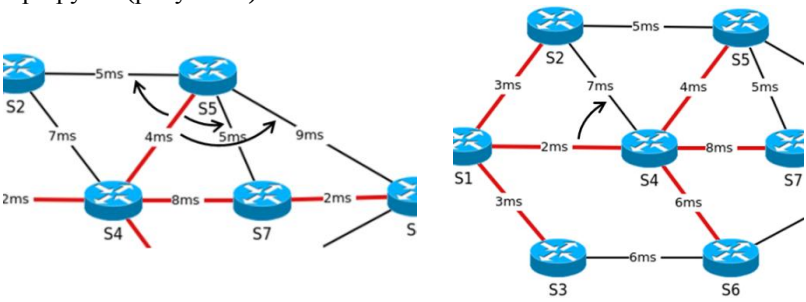
Теперь, определившись с типами и структурами необходимых переменных, можно перейти к самому алгоритму. Он будет состоять из четырех функций:

1) *set_pair_transition(switch: int)*;
2) *pair_transition_from_tree(v1: int, v2: int, new_v1: int)*;
3) *pair_transition_from_replacement(v1: int, v2: int, old_v1: int)*;
4) *run(v1: int, v2: int, w: float).*

Первая функция принимает номер коммутатора и определяет потенциальные парные переходы для канала связи (*parent[switch], switch*), находящегося в дереве оптимальных маршрутов. Основная ее задача – внести в *transitions*1 и *transitions*2 информацию о парных переходах.

Вначале в *transitions*1 создается пустой список с ключом (*parent[switch], switch*). Далее необходимо рассмотреть два случая: когда текущий коммутатор *switch* является листом дерева

оптимальных маршрутов, и когда не является. Если *switch* – лист дерева, то его родительский коммутатор может быть заменен на любой из смежных к *switch*. В противном случае, родительской коммутатор может быть заменен только теми коммутаторами, которые находятся на том же уровне или выше по иерархии в дереве оптимальных маршрутов (рисунок 4).



а) б)

Рисунок 4 – Потенциальные парные переходы канала S4 – S5 (а) и канала S1 – S4 (б)

Канал S1 – S4 может быть заменен только на канал S2 – S4. В противном случае могут возникнуть циклы.

Далее происходит просмотр всех смежных со *switch* коммутаторов *adjacent_sw*, с учетом вышерассмотренных случаев. Если родительской коммутатор может быть заменен на *adjacent_sw*, то вычисляется потенциальная точка выхода из дерева, а в уже созданный список с ключом (*parent[switch]*, *switch*) заносится пара ключ – значение $\{\omega^t$: *adjacent_sw*$\}$. В *transitions*2 также создается пустой список с ключом (adjacent_sw, *switch*). Вычисляется точка вхождения в дерево и в этот список добавляется пара $\{\omega^r$:*parrent[switch]*$\}$.

После выполнения этой функции для экспериментальной топологии словари *transitions*1 и *transitions*2 будут иметь следующий вид как в таблице 3.

Таблица 3. Значения словарей *transitions*1 и *transitions*2

Название	**Значение**
*transitions*1	{(None, 0): {}, (0, 1): {9: 3, 11: 4}, (0, 2): {14: 5}, (0, 3): {10: 1}, (3, 4): {6: 1, 13: 6, 19: 7}, (3, 5): {7: 2, 15: 7}, (3, 6): {9: 4}, (6, 7): {5: 4, 3: 5}}
*transitions*2	{(3, 1): {1: 0}, (4, 1): {-3: 0}, (5, 2): {-5: 0}, (1, 3): {-1: 0}, (1, 4): {3: 3}, (6, 4): {-4: 3}, (7, 4): {-6: 3}, (2, 5): {5: 3}, (7, 5): {-4: 3}, (4, 6): {4: 3}, (4, 7): {6: 6}, (5, 7): {4: 6}}

Далее рассмотрим функцию *run()*. В качестве аргументов она принимает два номера коммутаторов, и новое значение метрики канала связи (*v*1, *v*2). Во-первых, необходимо изменить значение метрики в исходной матрице смежности. Затем, следует определение, того, в каком словаре, *transitions*1 или *transitions*2, находится данный канал. Стоит обратить внимание на то, что, в словаре может ни оказаться такого канала-кортежа, но может оказаться (*v*2, *v*1). Такую ситуацию необходимо учитывать.

После этого рассматривается два случая: если канал имеется в *transitions*1 или если канал имеется в *transitions*2.

В первом случае, канал принадлежит дереву оптимальных маршрутов. Для него вычисляется точка выхода из дерева (минимальная из потенциальных) и определяется новый родительский коммутатор. Если новое значение метрики больше, чем точка выхода из дерева, то вызывается функция *pair_transition_from_tree(v*1, *v*2, *new_v*1*)*, которая производит парный переход.

Далее проверяется, является ли *v*2 листом дерева оптимальных маршрутов. Если нет, то необходимо проверить, нуждаются ли каналы связи, расположенные ниже по иерархии в дереве оптимальных маршрутов, в парных переходах. Для этого словарь *tag* сортируется по возрастанию значений. Для каждого коммутатора *v*3, значение метки которого больше или равно значению метки коммутатора *v*2, проверяется, нуждается ли канал (*parent[v*3*]*, *v*3) в парном переходе, с учетом того, что уже был выполнен переход с (*v*1, *v*2) на (*new_v*1, *v*2). Если переход необходим, то действия аналогичны тем, которые производились при первом парном переходе.

Теперь пересчитывается длина пути от *src* до *v*2. Также обновляется информация о парных переходах с помощью функции *set_pair_transition()*.

Во втором случае, когда канал (*v*1, *v*2) имеется в *transitions*2, новое значение метрики сравнивается с точкой входа в дерево. Если новое значение меньше, то определяется коммутатор *old_v*1, который будет заменен на *v*1 и вызывается функция *pair_transition_from_ replacement(v*1, *v*2, *old_v*1*)*. Она совершит парный переход каналов (*v*1, *v*2) и (*old_v*1, *v*2).

Если *v*2 не является листом дерева оптимальных маршрутов, то снова необходимо проверить, нуждаются ли в парных переходах каналы, но уже расположенные выше по иерархии в дереве оптимальных маршрутов, чем текущий. Это делается аналогично предыдущему случаю, но список *tag* сортируется по убыванию значений.

После этого пересчитывается длина пути от *src* до *v*2 и обновляется информация о парных переходах с помощью функции *set_pair_transition()*.

Функция *run()* описывает основную логику работы алгоритма. В ней используются такие функции как *pair_transition_from_tree()* и *pair_transition_from_replacement()*, производящие парные переходы.

Первая вызывается в случае, когда увеличивается метрики канала связи, находящегося в дереве оптимальных маршрутов. Она принимает номера начального *v*1 и конечного *v*2 коммутаторов канала (*v*1, *v*2), а также новый начальный коммутатор *new_v*1. Суть функции заключается в обновлении информации в переменных, представленных в табл. 1. Так, из *min_tree* необходимо удалить (*v*1, *v*2) и внести (*new_v*1, *v*2), а из *links*, наоборот, удалить (*new_v*1, *v*2) и внести (*v*1, *v*2). Также удаляются соответствующие ключи из *transitions*1 и *transitions*2. Далее, обновляется путь до *v*2, с учетом перехода, его коммутатор-родитель и значение метки.

Вторая функция вызывается в случае, когда уменьшается метрика канала связи, не находящегося в дереве оптимальных маршрутов. Она принимает номера начального *v*1 и конечного *v*2 коммутаторов канала (*v*1, *v*2), а также номер коммутатора *old_v*1, который будет заменен на *v*1. Ее действия аналогичны предыдущей функции.

Запуск алгоритма парных переходов производится следующим образом:

1) создание и заполнение переменных (таблица 1);

2) получение информации о парных переходах с помощью функции *set_pair_transition();*

3) получение информации об изменении метрики канала связи;

4) запуск функции *run()*;

5) возвращение к шагу 3.

В работе рассмотрены особенности реализации алгоритма парных переходов в ПКС на языке Python. Описаны переменные и функции, необходимые для работы алгоритма.

Работа выполнена при финансовой поддержке гранта Президента РФ МД-3201.2022.1.6.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Dijkstra E.W. A note on two problems in connexion with graphs // Numer. Math / F. Brezzi – Springer Science+Business Media, 1959. – Vol. 1, Iss. 1. – Pp. 269–271.

2. Bellman R. On a Routing Problem // Quarterly of Applied Mathematics. – 1958. – Vol 16, No. 1. – Pp. 87-90.

3. Floyd R.W. Algorithm 97: Shortest path. – Commun. ACM 5, 6. – 1962. – 345 p.

4. Перепелкин Д.А., Перепелкин А.И. Разработка алгоритмов адаптивной маршрутизации в корпоративных вычислительных сетях // Вестник Рязанской государственной радиотехнической академии. – 2006. – № 19. – С. 114-116.

5. Корячко В.П., Перепелкин Д.А. Программно-конфигурируемые сети. Учебник для вузов. – М.: Горячая линия-Телеком, 2020. – 288 с.

УДК 004.72

Д.А. ПЕРЕПЕЛКИН, Г.А. ЗАВАЛИШИН
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ПРОГРАММНО-КОНФИГУРИРУЕМЫЙ ИНТЕРНЕТ ВЕЩЕЙ

Рассмотрена архитектура программно-конфигурируемого Интернета вещей.

Традиционная сетевая инфраструктура состоит из различных сетевых устройств, таких как коммутаторы, маршрутизаторы и промежуточные устройства, в которых установлены специализированные интегральные схемы для выполнения специализированных задач. Поэтому устройства предварительно запрограммированы с различными сложными правилами (т.е. протоколами), которые не могут быть изменены в режиме реального времени для выполнения выделенных задач.

Кроме того, из-за ограниченных ресурсов устройств они не могут быть предварительно запрограммированы с множество правил для предоставления оптимальных сетевых услуг. Следовательно, традиционные сетевые технологии неспособны адаптировать адекватные политики для удовлетворения специфических требований Интернета вещей в режиме реального времени. Для устранения таких ограничений в традиционных сетях предлагается новая концепция, известная как программная-конфигурируемые сети (ПКС).

ПКС – это развивающаяся сетевая архитектура, с помощью которой управление сетью может быть отделено от традиционных аппаратных устройств. Таким образом, основная цель ПКС состоит в том, чтобы отделить плоскость управления от плоскости данных,

включающей устройства пересылки. В результате на физических устройствах может быть реализована адекватная логика управления в зависимости от требований конкретного приложения в режиме реального времени.

Интернет вещей не является средством для полной автоматизации своих компонентов. Он предполагает предоставление доступа человека к вещам. В IoT каждая вещь имеет свой уникальный идентификатор, которые совместно образуют континуум вещей, способных взаимодействовать друг с другом, создавая временные или постоянные сети.

Концепция интернета вещей (IoT) позволяет физическим объектам, в которые встроены датчики, исполнительные механизмы и сетевые подключения, собирать данные и обмениваться ими между собой на основе сотрудничества. Следовательно, очевидно, что ожидается, что беспроводные сети будут играть ключевую роль в мониторинге различных объектов с целью создания «умных» вещей (таких как «умный дом», «интеллектуальные транспортные системы» и «умное здравоохранение»). Одновременно ПКС использует парадигму централизованного управления сетью, разделяя логику управления с физических устройств на реальной сетевой платформе. Таким образом, традиционные устройства пересылки логически разделены на две плоскости – плоскость управления и плоскость данных. Различные операции управления сетью выполняются через плоскость управления, тогда как плоскость данных относится к физическим устройствам в сети.

Далее представлены некоторые ключевые требования приложений Интернета вещей, которые потенциально могут быть выполнены с помощью технологий ПКС для реализации концепции программно-конфигурируемого Интернета вещей.

- Управление сетью: ожидается, что примерно через десятилетие миллиарды устройств будут использоваться во всем мире благодаря технологии Интернета вещей. Таким образом, очевидно, что с устройств будет генерироваться огромный объем данных, которые необходимо своевременно и эффективно обрабатывать. Следовательно, сетевое управление является важным фактором для управления такими огромная коллекция устройств и огромная информация, генерируемая ими. Такие требования могут быть выполнены с помощью технологии на основе ПКС, поскольку она централизованно использует глобальное представление о сети.
- Виртуализация сетевых функций: заранее запрограммированный характер традиционных сетевых технологий не

позволяет устройствам выполнять множество задач, хотя они способны это делать. Поэтому требуется виртуализировать функции устройств и изменять их в режиме реального времени. Недавно была представлена концепция виртуализации сетевых функций (NFV), которая позволяет устройствам выполнять множество задач, изменяя при этом свои функции в режиме реального времени в зависимости от требований конкретного приложения.

- Доступ к информации из любого места: Как обсуждалось в вышеприведенных пунктах, предполагается, что в IoT будут подключены миллиарды устройств. Кроме того, владельцы устройств должны иметь доступ к ним из любого места и в любое время, чтобы они могли контролировать и изменять функции своих устройств, в зависимости от требований, плавным способом. Управлять такими устройствами в сети можно с помощью технологий на основе SDN, сохраняя при этом конфиденциальность других.
- Использование ресурсов: Недостаточное или чрезмерное использование может снизить производительность сети, что, в свою очередь, сводит к минимуму полезность сети. Следовательно, эффективное сопоставление. Таким образом, устройства в центре обработки данных могут динамически включаться/выключаться в зависимости от требований, что, в свою очередь, создает энергоэффективную сеть центра обработки данных.
- Безопасность и конфиденциальность: Наконец, защита устройств и сети является важным фактором, позволяющим нескольким устройствам, поставщикам и пользователям участвовать в единой платформе.

Архитектура управления ПКС

Мы следуем обобщенной архитектуре ПКС, которая состоит из уровней инфраструктуры, управления и приложений. На уровне инфраструктуры развертываются сенсорные узлы и устройства доступа. Сенсорные узлы отправляют полученную информацию в устройства доступа, а устройства доступа пересылает информацию в центр обработки данных для вычислений. Используя концепцию ПКС в беспроводной сети, сенсорными узлами можно управлять централизованно, в зависимости от требований конкретного приложения. Следовательно, контроллер принимает адекватные решения и контролирует всю сеть.

Архитектуру ПКС можно представить как множество программ и аппаратных коммутаторов, подключенных к общему контроллеру OpenFlow и реализованных в виде обычного сервера со специальным ПО. Такая архитектура состоит из 3 уровней:

1. Инфраструктурный уровень (ИУ), состоящий из набора сетевых устройств (коммутаторов и каналов передачи данных).

2. Уровень управления (УУ), включающий в себя сетевую операционный систему, которая обеспечивает приложениям сетевые сервисы и программный интерфейс для управления сетевыми ресурсами и сетью.

3. Уровень сетевых сервисов (УСС), включающий приложения, позволяющие более гибко и эффективно управлять сетью.

Однако добавляется ещё один уровень. Это уровень сети устройств Интернета вещей (рисунок 1).

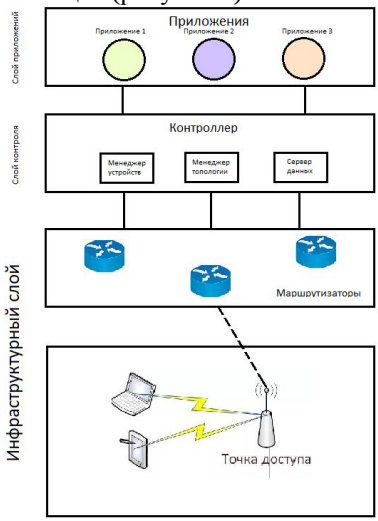


Рисунок 1 – Архитектура ПКС Интернета вещей

В статье представлен подробный обзор существующих технологии на основе ПКС в контексте приложений Интернета вещей, чтобы обеспечить бесперебойное, экономически эффективное и надежное предоставление услуг пользователям. Были определены некоторые важные технические проблемы и представили несколько будущих направлений исследований для их решения.

Работа выполнена при финансовой поддержке гранта Президента РФ МД-3201.2022.1.6.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Software-Defined Networking (SDN) Definition. – Текст: электронный. – URL: https://cse.iitkgp.ac.in/~smisra/theme_pages/sdn/index.html (дата обращения: 15.04.2023).

2. Сетевые технологии Интернета вещей. – Текст: электронный. – URL: https://habr.com/ru/company/ericsson_ru/blog/301494/ (дата обращения: 15.04.2023).

УДК 004.42

Д.А. ПЕРЕПЕЛКИН, Г.А. ЗАВАЛИШИН
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ ПРИЛОЖЕНИЯ ФУДШЕРИНГА

Рассматривается разработка серверной части приложения фудшеринга. Используются такие технологии как Spring Boot, Docker, Swagger, для администрирования Selectel.

Целью создания данного приложения – дать людям возможность не выбрасывать продукты, а делиться с теми, кто в этом нуждается.

В настоящее время у людей остаётся лишняя еда, которая зачастую просто портиться и выбрасывается на помойку. Мы помогаем людям найти бесплатную еду, а также тем, кто не хочет выбрасывать хорошие продукты или просто хочет сделать доброе дело и поделиться продуктами с теми, кто в них нуждается.

Данный сервис позволит найти в ближайшей доступности нужные продукты или добавить предложение о тех продуктах, срок годности которых ещё не истёк, но который по разным причинам стали не нужны.

Для разработки REST API на Java использовался Spring Boot. Spring Boot – это полезный проект, целью которого является упрощение создания приложений на основе Spring. Он позволяет наиболее простым способом создать web-приложение, требуя от разработчиков минимум усилий по его настройке и написанию кода. Spring Boot обладает большим функционалом, но его наиболее значимыми особенностями являются: управление зависимостями, автоматическая конфигурация и встроенные контейнеры сервлетов.

Были созданы следующие сущности:

- AbstractEntity (Абстрактная);
- Ad (Объявление);
- Address (Адрес);
- AdOrder (Заказ);
- AdReview (Отзыв);

- AdStatus (Статус объявления);
- Category (Категория);
- City (Город);
- Image (Изображение);
- OrderStatus (Статус заказа);
- Sku (Единицы измерения);
- SmsCode (код подтверждения);
- Subway (Метро);
- User (Пользователь);
- UserRole (Роль пользователя);

Следующий этап – это создание ресурсов REST. Для данных сущностей были разработаны соответствующие контроллеры для обработки запросов и ответов с сервера.

Контроллеры получают данные от пользователя приложения, и конкретный метод посылает ответ этому пользователю. Методы контроллеров реализованы в файлах сервиса приложения. Они также есть у каждой сущности.

В этих файлах прописана логика работы каждого конкретного метода (добавления, обновления, удаления, получения данных):

- GET – получение конкретного ресурса (по id) или коллекцию ресурсов;
- POST – создание нового ресурса;
- PUT – обновление конкретного ресурса (по id);
- DELETE – удаление конкретного ресурса по id.

Для корректности работы запросов использовался Swagger. Swagger – это фреймворк для спецификации RESTful API. Его прелесть заключается в том, что он дает возможность не только интерактивно просматривать спецификацию, но и отправлять запросы – так называемый Swagger UI. Его часть представлена на рисунке 1.

Для контейнеризации приложения была выбрана Docker технология.

Docker – это технология контейнеризации, которая позволяет упаковывать приложения и их зависимости в легковесные контейнеры, которые могут быть развернуты на любой машине с Docker. Контейнеры Docker изолируют приложение и его зависимости от операционной системы хоста, что обеспечивает большую гибкость и надежность.

Docker использует ядро операционной системы хоста для запуска контейнеров, что позволяет использовать ресурсы машины более эффективно. Контейнеры Docker являются легковесными и

быстрыми, что обеспечивает быстрое развертывание и масштабирование приложений.

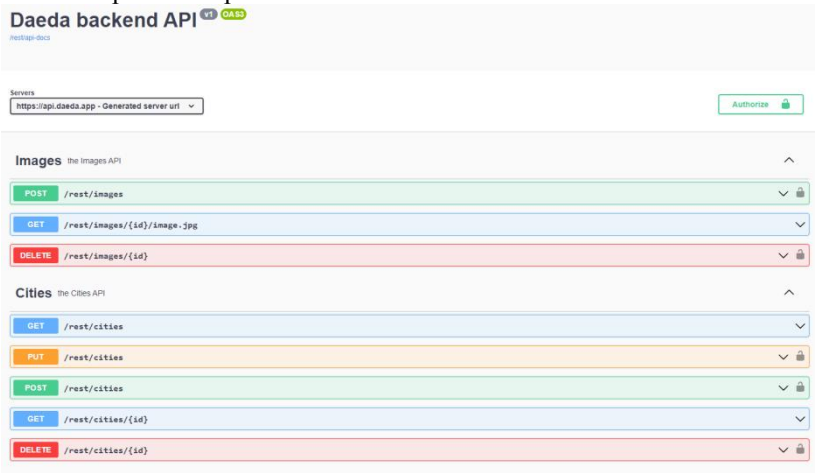


Рисунок 1 – Swagger UI

Основными компонентами Docker являются Docker Engine – основной компонент для управления контейнерами, Docker Hub – репозиторий для хранения и распространения образов контейнеров, а также Docker Compose – инструмент для описания и запуска множества контейнеров как единого приложения.

Для администрирования была выбрана такая платформа как Selectel. На данный момент создано два сервера: для ветки разработки и для ветки продакшена, то, что доступно пользователям на данный момент (рисунок 2).

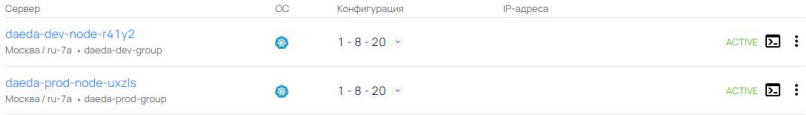


Рисунок 2 – Сервера приложения

В данной платформе организовано администрирование кластеров Kubernetes.

Как минимум, кластер содержит плоскость управления и одну или несколько вычислительных машин или узлов. Плоскость управления отвечает за поддержание желаемого состояния кластера, например, за то, какие приложения запущены и какие образы контейнеров они используют.

Кластер является основой ключевого преимущества Kubernetes: возможность планировать и запускать контейнеры на группе компьютеров, будь то физические или виртуальные, локально или в облаке. Контейнеры Kubernetes не привязаны к отдельным машинам.

Также при разработке использовалась система контроля версий GitLab и Agile доска Jira. Данные средства упрощали процесс работы в команде и позволяли быстро и качественно распределять задачи между напарниками, что ускоряло процесс разработки.

В данной статье описана разработка и поддержка серверной части приложения для фудшеринга с использованием таких технологий как Spring Boot, Swagger, Docker, Selectel. Данное приложение является актуальной разработкой в наше время и теперь у людей появилась возможность удобно размещать объявления и делиться продуктами с нуждающимися.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Selectel – облачные инфраструктурные сервисы и услуги дата-центров. – Текст: электронный. – URL: https://selectel.ru/ (дата обращения: 15.04.2023).

2. Spring Bott. – Текст: электронный. – URL: https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/ (дата обращения: 15.04.2023).

3. Docker. – Текст: электронный. – URL: https://docs.docker.com/ (дата обращения: 15.04.2023).

УДК 004.7

Д.А. ПЕРЕПЕЛКИН, Д.Д. ТКАЧЕВ
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

РАЗРАБОТКА УСТРОЙСТВА СБОРА ПАРАМЕТРОВ ДЛЯ СИСТЕМЫ ИНТЕРНЕТА ВЕЩЕЙ

Рассматривается разработка устройства сбора параметров для системы Интернета вещей на основе микроконтроллера ESP32, датчика температуры и влажности воздуха DHT11, фоторезистора GL5537 и датчика движения HC-SR501.

В настоящее время сбор и анализ параметров окружающей среды является важной задачей для многих областей, таких как экология, метеорология, сельское хозяйство. Развитие технологии Интернета вещей позволяет создавать устройства, собирающие данные

о параметрах окружающей среды и отправляющие их на удаленный сервер (облачную платформу) для дальнейшего анализа.

Устройство сбора параметров для системы интернета вещей

Устройство сбора параметров для системы Интернета вещей разработано на основе микроконтроллера ESP32 и подключенных к нему датчиков, позволяющих получать данные о температуре, влажности воздуха, уровне освещенности и наличии движения.

ESP32 является мощным микроконтроллером с двумя ядрами процессора, поддержкой Wi-Fi и Bluetooth, а также различными интерфейсами для подключения периферийных устройств. Благодаря высокой производительности, низкой цене, поддержке различных протоколов передачи данных и широкой периферии микроконтроллер ESP32 является идеальным выбором для разработки устройств, выполняющих сбор и передачу данных из окружающей среды.

Для сбора данных о температуре и влажности воздуха используется датчик DHT11. Данный датчик имеет аналого-цифровой преобразователь, позволяющий преобразовать аналоговые значения влажности и температуры воздуха в цифровые, а также он имеет простой интерфейс и может быть легко подключен к микроконтроллеру ESP32. Несмотря на то, что датчик DHT11 не обладает высоким быстродействием и точностью работы, он недорог и прост в использовании, поэтому хорошо подходит для контроля значений температуры и влажности воздуха окружающей среды. Пины VCC и GND датчика температуры и влажности воздуха DHT11 подключены к соответствующим пинам ESP32, а сигнальный пин датчика подключен к цифровому пину 14 микроконтроллера ESP32.

Для измерения уровня освещенности окружающей среды используется фоторезистор GL5537. Фоторезистор GL5537 – это полупроводниковый элемент, который изменяет свое сопротивление в зависимости от интенсивности света, попадающего на его поверхность. Данный фоторезистор был выбран за счет своей компактности, низкой цены и небольшой потребляемой мощности. Один конец фоторезистора GL5537 подключен к пину VCC, другой конец к пину GND, а середина фоторезистора подключена к аналоговому пину 35 ESP32.

Датчик HC-SR501 используется для обнаружения движения. Данный датчик движения использует инфракрасную технологию для обнаружения движения в зоне обзора. При достаточно низкой цене он обладает широким углом обзора, большой зоной обнаружения и низким энергопотреблением. Пины VCC и GND датчика движения HC-SR501 подключены к соответствующим пинам ESP32, а

сигнальный пин подключен к цифровому пину 27 микроконтроллера ESP32.

Для программирования микроконтроллера ESP32 используется язык Arduino. Листинг кода программы для сбора данных о температуре, влажности, уровне освещенности окружающей среды и наличии движения, а также отправки полученных данных на удаленный сервер или облачную платформу представлен ниже.

```
#include "DHT.h"
#include <WiFi.h>
#include <ArduinoJson.h>
#define DHTPIN 14
#define DHTTYPE DHT11
#define LIGHTPIN 35
#define MOTIONPIN 27

const char* ssid = " SSID";
const char* password = " PASSWORD";
const uint16_t port = 7842;
const char * host = " HOST";

DHT dht(DHTPIN, DHTTYPE);
WiFiClient client;

void setup() {
  Serial.begin(115200);
  pinMode(LIGHTPIN, INPUT);
  pinMode(MOTIONPIN, INPUT);
  dht.begin();

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");
  while (!client.connect(host, port)) {
   Serial.println("Connection to host failed");
   delay(5000);
  }
  Serial.println("Host connected");
}

void loop() {
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();
  int light = analogRead(LIGHTPIN);
```

```
  int motion = digitalRead(MOTIONPIN);

  if (client.connected()) {
    DynamicJsonDocument doc(2048);
    doc["HUMIDITY"] = humidity;
    doc["TEMPERATURE"] = temperature;
    doc["MOTION"] = motion;
    doc["LIGHT"] = light;
    char json[400];
    serializeJson(doc, json);
    client.print(json);
  }
  else {
    client.stop();
    while (!client.connect(host, port)) {
      Serial.println("Connection to host failed");
      delay(5000);
    }
    Serial.println("Host reconnected");
  }
  delay(500);
}
```

В приведенном листинге сначала инициализируются все датчики и задаются пины, к которым они подключены. Затем осуществляется подключение микроконтроллера ESP32 к сети Wi-Fi и установление соединение с удаленным сервером, позволяющим собирать и анализировать данные о параметрах окружающей среды. После чего создается бесконечный цикл, который собирает данные с каждого датчика и отправляет их на удаленный сервер. Если соединение с сервером по какой-либо причине будет разорвано, микроконтроллер циклически будет осуществлять попытки установить новое соединение.

В статье разработано устройство сбора параметров для системы Интернета вещей на основе микроконтроллера ESP32, датчика температуры и влажности воздуха DHT11, фоторезистора GL5537 и датчика движения HC-SR501. Это устройство может быть использовано для мониторинга параметров окружающей среды в режиме реального времени внутри помещения, на улице или в других условиях. Данные можно передавать на удаленный сервер для дальнейшего анализа и принятия решений.

Работа выполнена при финансовой поддержке гранта Президента РФ МД-3201.2022.1.6.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. ESP32 Wi-Fi and Bluetooth MCU / Espressif System. – Текст: электронный. – URL: https://www.espressif.com/en/products/socs/esp32 (дата обращения: 15.04.2023).
2. DHT11, DHT22 and AM2302 Sensors. – Текст: электронный. – URL: https://learn/adafruit/com/dht (дата обращения: 15.04.2023).
3. Фоторезистор GL 5537. – Текст: электронный. – URL: https://voltiq.ru/shop/fotorezistor-gl-5537-datchik-osveshhennosti/ (дата обращения: 15.04.2023).
4. Инфракрасный датчик движения HC-SR501. – Текст: электронный. – URL: https://robotchip.ru/obzor-infrakrasnogo-datchika-dvizheniya-hc-sr501/ (дата обращения: 15.04.2023).
5. Примеры проектов на ESP32. – Текст: электронный. – URL: https://randomnerdtutorials.com/projects-esp32/ (дата обращения: 15.04.2023).

УДК 004.72

Д.А. ПЕРЕПЕЛКИН, Д.Д. ТКАЧЕВ
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ЧЕТЫРЕХУРОВНЕВАЯ АРХИТЕКТУРА ПРОГРАММНО-КОНФИГУРИРУЕМОЙ СЕТИ УСТРОЙСТВ ИНТЕРНЕТА ВЕЩЕЙ

Предложена четырехуровневая архитектура программно-конфигурируемой сети устройств Интернета вещей и рассмотрены области ее применения.

В последние годы концепция Интернета вещей (IoT) становится все более популярной и широко используется в различных областях, включая промышленность, здравоохранение, сельское хозяйство и транспорт. Однако, с ростом количества устройств, подключенных к сети Интернета вещей, возникает проблема управления и контроля за этими устройствами. Поэтому для эффективного использования систем на основе концепции Интернета вещей необходимо иметь архитектуру, которая бы позволяла работать с большим количеством устройств и обрабатывать большие объемы данных в режиме реального времени.

Четырехуровневая архитектура ПКС устройств интернета вещей

Четырехуровневая архитектура программно-конфигурируемой сети устройств Интернета вещей (рисунок 1) представляет собой решение для управления и контроля над системами IoT-устройств.

Первый уровень данной архитектуры – это уровень устройств. На этом уровне располагаются все устройства, которые подключены к сети Интернета вещей, такие как датчики, контроллеры и различные механизмы. Основной задачей устройств на этом уровне является сбор данных и их передача на следующий уровень с помощью протоколов передачи данных, таких как Wi-Fi, Bluetooth, Zigbee.

Вторым уровнем в архитектуре программно-конфигурируемой сети устройств Интернета вещей является уровень сети, который играет важную роль в обеспечении связи между устройствами и центральным сервером за счет использования различных протоколов связи. Для обеспечения безопасности передачи данных на уровне сети могут использоваться различные методы шифрования и аутентификации, такие как SSL/TLS, WPA2, EAP, которые обеспечивают защиту от несанкционированного доступа к данным и перехвата информации.

Третий уровень – это уровень управления и обработки данных. Он включает в себя облачные сервисы и серверы, которые обрабатывают данные, полученные от устройств на первом уровне. Основной задачей этого уровня является обеспечение анализа данных и принятие решений на основе этих данных. Для этого используются различные аналитические инструменты, такие как машинное обучение, статистический анализ, моделирование данных. Данный уровень позволяет также настраивать параметры системы, определять пороги тревоги для определенных событий или настраивать автоматические действия в ответ на определенные внешние условия.

Четвертым уровнем является уровень приложений. На этом уровне создаются различные программные продукты, включая мобильные приложения, веб-приложения, программное обеспечение для ПК, которые позволяют пользователям управлять устройствами Интернета вещей, получать данные и анализировать их.

Все уровни данной архитектуры программно-конфигурируемой сети устройств Интернета вещей взаимодействуют между собой, обмениваясь данными и выполняя задачи. Это позволяет создавать интеллектуальные системы, которые могут автоматически реагировать на изменения в окружающей среде и в режиме реального времени принимать решения на основе обработки и анализа данных.

Преимуществом данной архитектуры является ее гибкость и масштабируемость. Каждый уровень может быть расширен или изменен без влияния на другие уровни. Использование программно-конфигурируемой сети позволяет быстро адаптироваться к различным изменениям в IoT-системе и увеличивать ее производительность.

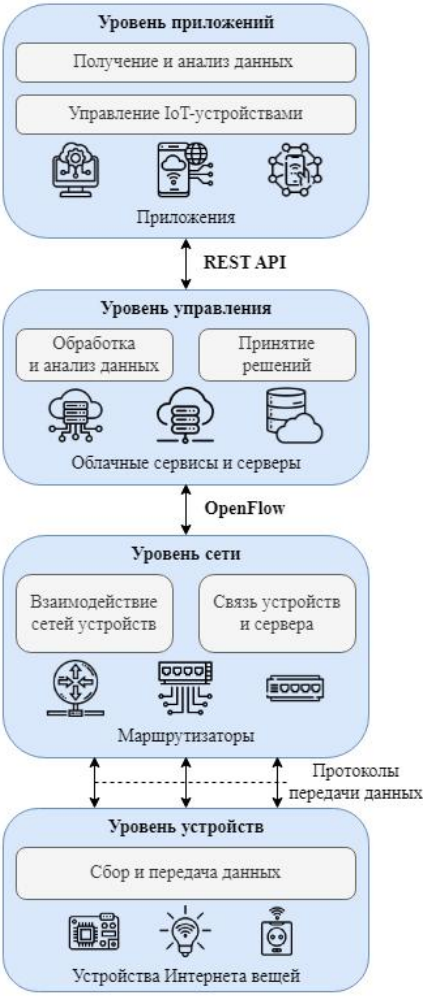


Рисунок 7 – Четырехуровневая архитектура программно-конфигурируемой сети устройств Интернета вещей

Применение четырехуровневой архитектуры ПКС устройств интернета вещей

Программно-конфигурируемая сеть устройств Интернета вещей может быть применена в различных областях, таких как:

1. Управление зданиями: IoT-устройства могут использоваться для управления освещением, температурой, вентиляцией и другими

системами в зданиях. Например, датчики температуры и влажности воздуха могут использоваться для автоматического регулирования температуры и влажности в помещениях.

2. Мониторинг окружающей среды: IoT-устройства могут использоваться для мониторинга качества воздуха, уровня шума, загрязнения воды и других параметров окружающей среды. Эти данные могут быть использованы для принятия решений по улучшению экологической ситуации.

3. Умный город: IoT-устройства могут использоваться для управления транспортной инфраструктурой, уличным освещением, мусоропроводами и другими системами в городах. Например, датчики движения могут использоваться для автоматического включения и выключения уличного освещения.

4. Промышленность: IoT-устройства могут использоваться для мониторинга и управления производственными процессами, оборудованием и инфраструктурой. Например, датчики давления и температуры могут использоваться для мониторинга состояния оборудования и предотвращения аварий.

5. Здравоохранение: IoT-устройства могут использоваться для мониторинга здоровья пациентов, управления оборудованием и обеспечения безопасности в медицинских учреждениях. Например, датчики сердечного ритма могут использоваться для мониторинга состояния пациентов в реальном времени.

В статье предложена четырехуровневая архитектура программно-конфигурируемой сети устройств Интернета вещей, которая может быть основой для создания интеллектуальных систем, способных автоматически реагировать на изменения в окружающей среде и выполнять задачи без участия человека. Данная архитектура также обеспечивает гибкость, масштабируемость и безопасность сети.

Работа выполнена при финансовой поддержке гранта Президента РФ МД-3201.2022.1.6.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Deshpande A., Jha S.K., Jena S.K. Internet of Things: architecture, applications, and challenges / Internet of Things and Big Data Analytics Toward Next-Generation Intelligence. – Springer, 2018.

2. Shakir M.Z., Ali M.A., Alam M.S. A comprehensive review on the internet of things (IoT) architecture, applications, security issues, and challenges / Internet of Things (IoT): Technologies, Applications and Implementations. – CRC Press, 2019.

УДК 004.65

И.В. ПЕЧЕНИН
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ДОПОЛНЕННАЯ И ВИРТУАЛЬНАЯ РЕАЛЬНОСТЬ В ОБРАЗОВАНИИ И IT

Рассматриваются основные направления использования дополненной и виртуальной реальности, примеры внедрения, преимущества и недостатки.

Современные технологии, такие как дополненная и виртуальная реальность, имеют огромный потенциал для изменения образовательной среды. Они позволяют ученикам получать новые знания и навыки в интерактивной и привлекательной форме.

AR и VR могут быть использованы для создания интерактивных уроков и тренировок, а также для создания иммерсивных образовательных сценариев.

Тем не менее, AR и VR могут стать незаменимыми инструментами в образовании, помогая создавать более интерактивное, привлекательное и эффективное обучение.

Однако, несмотря на все преимущества, эти технологии могут быть дорогостоящими для внедрения в образовательную среду, а также существует риск того, что ученики могут перестать обращать внимание на реальный мир.

Что такое AR и VR, какие преимущества и недостатки они имеют

Дополненная реальность (AR) – это технология, которая позволяет добавлять визуальные элементы к реальному миру, используя мобильные устройства или специальные очки. Она может быть использована для создания интерактивных уроков и тренировок. Например, анатомические модели могут быть дополнены визуальными элементами, чтобы показать, как работают различные органы в теле человека.

Виртуальная реальность (VR) – это технология, которая создает полностью искусственную среду, которая может быть идентична реальному миру или абсолютно фантастической. Она может быть использована для создания образовательных сценариев. Например, ученики могут посетить древний Рим или погрузиться в глубины океана, чтобы изучать экосистему.

Преимущества использования AR и VR в образовании очевидны. Они могут помочь ученикам получить лучшее понимание сложных

концепций и улучшить их способности к принятию решений. Они также могут помочь снизить стоимость и повысить эффективность обучения, так как они позволяют ученикам учиться без необходимости посещать дорогостоящие места, такие как музей и лаборатории.

Недостатки использования AR и VR в образовании также существуют. Некоторые преподаватели могут не иметь достаточно опыта или знаний, чтобы использовать эти технологии в своих уроках, а также такой класс оборудования является довольно дорогостоящим.

Тем не менее, если эти риски будут управляться и преодолены, то AR и VR могут стать незаменимыми инструментами в образовании. Они могут помочь создать более интерактивное, привлекательное и эффективное обучение, что может улучшить успеваемость и мотивацию учеников.

Использование AR и VR в образовательной сфере IT-технологий и примеры внедрения

Дополненная и виртуальная реальность могут быть особенно полезны в обучении IT-технологиям. Например, они могут использоваться для создания иммерсивных симуляторов, которые помогают ученикам разобраться в сложных процессах и принципах программирования, дизайна веб-сайтов или создания приложений.

AR и VR могут использоваться для создания интерактивных курсов по кибербезопасности или машинному обучению. Все это помогает ученикам получать практические навыки и опыт, что в свою очередь повышает эффективность обучения и подготавливает их к работе в современной IT-индустрии.

Одним из примеров внедрения AR и VR в обучении IT-технологиям является платформа Unity Learn, которая предоставляет доступ к бесплатным обучающим ресурсам и иммерсивным симуляторам для разработчиков игр и приложений. С помощью VR-гарнитуры Oculus Rift, пользователи могут получить практический опыт создания игр и приложений в виртуальной среде.

Другой пример – платформа Codecademy, которая предоставляет интерактивные курсы по программированию на различных языках программирования. С помощью AR-технологии пользователи могут сканировать код на экране компьютера или монитора и получать дополнительную информацию о каждой строке кода, что помогает им лучше понимать процессы программирования.

AR и VR могут использоваться в обучении машинному обучению. Например, платформа TensorFlow предоставляет доступ к обучающим модулям и симуляторам, которые помогают ученикам понять, как работает машинное обучение, и как использовать

TensorFlow для создания своих собственных моделей машинного обучения.

Внедрение технологий дополненной и виртуальной реальности в образовательную сферу IT России может быть полезно как для студентов, так и для преподавателей. Они могут использовать технологии для создания более интересных и эффективных учебных материалов, а студенты могут получить доступ к новым знаниям и навыкам.

Примером использования технологий дополненной и виртуальной реальности в образовательной сфере IT России может быть проект «Виртуальная лаборатория» от компании Intel. Этот проект представляет собой симулятор, который позволяет студентам практиковаться в решении задач и экспериментах без необходимости использовать физические объекты и оборудование.

Таким образом, дополненная и виртуальная реальность могут быть эффективным инструментом для обучения IT-технологиям, и уже сегодня они активно используются в различных образовательных проектах.

В заключении можно отметить, что AR и VR являются перспективными технологиями для обучения в сфере IT и образования в целом.

Они позволяют создавать иммерсивные симуляторы, интерактивные курсы и обучающие модули, которые помогают ученикам лучше понимать и запоминать информацию. Кроме того, использование AR и VR позволяет создавать безопасные условия для экспериментов и практических занятий, что особенно важно в случае обучения программированию и машинному обучению.

В целом, AR и VR открывают новые возможности для обучения IT-технологиям и помогают студентам лучше подготовиться к будущей карьере в этой области.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Киреева Ю.В. Возможности и проблемы использования технологий дополненной и виртуальной реальности в образовательном процессе / Ю.В Киреева // Вестник Томского государственного университета. – 2017. – № 421. – С. 106-114 с.

2. Кузнецова М.Ю. Виртуальная реальность в образовании: применение технологии в педагогической практике / М.Ю. Кузнецова // Инновации в образовании. – 2016. – № 2 (16). – С. 77-84.

3. Николаева Е.А. Применение технологий дополненной и виртуальной реальности в разработке программного обеспечения / Е.А

Николаева, Е.А Комарова // Научно-технический вестник информационных технологий, механики и оптики. – 2018. – № 5. – С. 110-115.

4. Гребенщикова Е.А. Применение технологий дополненной и виртуальной реальности в сфере IT России / Е.А Гребенщикова, Е.В Шевченко // Современные проблемы науки и образования. – 2018. – № 2. – С. 26-32.

УДК 004.65

И.В. ПЕЧЕНИН
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ПЕРСПЕКТИВЫ ВНЕДРЕНИЯ НЕЙРОННЫХ СЕТЕЙ В РЕАЛИЗАЦИЮ СИСТЕМ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ

Рассматриваются основные направления использования нейронных сетей, примеры внедрения в реализацию систем поддержки принятия решений (СППР), а также преимущества и недостатки интеграции нейронных сетей в СППР.

Нейронные сети являются мощным инструментом в области искусственного интеллекта и могут быть использованы для реализации систем поддержки принятия решений принятия решений. СППР используются в различных областях и помогают принимать решения на основе анализа данных, учитывая различные факторы.

Внедрение нейронных сетей в СППР может значительно улучшить качество принимаемых решений, позволяя обрабатывать большие объемы данных и выдавать точные прогнозы.

Однако, необходимо учитывать сложность и требования к обучению и поддержанию модели.

Определение нейронной сети и интеграция с системой поддержки принятия решений, примеры внедрения

Нейронная сеть – это математическая модель, которая имитирует работу нервной системы человека и может использоваться для решения сложных задач в различных областях. Она состоит из множества взаимосвязанных элементов (нейронов), которые обрабатывают входные данные и выдают результаты.

Нейронные сети способны обучаться на основе большого количества данных и выдавать точные прогнозы, что делает их идеальным инструментом для реализации систем поддержки принятия решений.

Системы поддержки принятия решений (СППР) используются в различных областях, включая финансы, здравоохранение,

производство и многие другие. Они помогают принимать различные решения, такие как риски, потенциальные выгоды и ограничения.

Например, в финансовой отрасли нейронные сети могут использоваться для прогнозирования цена на акции, определения рисков и оценки потенциальных выгод, определения оптимальной стратегии финансирования или автоматического определения рисковых инвестиционных портфелей. В здравоохранении нейронные сети могут помочь в диагностике заболеваний и определении эффективности лечения. Также нейронные сети могут быть использованы в системах управления производством для автоматизации процесса контроля за качеством продукции. Например, нейронные сети могут быть обучены распознаванию дефектов на производственной линии.

Другой пример использования нейронных сетей – это автоматизация процесса классификации и анализа больших объемов данных в IT-отрасли. Например, нейронные сети могут быть использованы для классификации текстовых данных (например, для автоматического определения тематики новостных статей), для анализа отзывов пользователей о продукте или для определения мнения клиентов о качестве обслуживания. Также нейронные сети могут принимать решения для улучшения производительности баз данных или для оптимизации ресурсов в виртуальных средах на основе исходных данных.

Однако, необходимо учитывать, что нейронные сети требуют больших объемов данных для обучения и поддержания точности прогнозов. Также необходимо проводить мониторинг и обновление модели, чтобы учитывать изменения в данных и среде.

В целом, нейронные сети могут быть использованы в различных областях и IT-отраслях для автоматизации процессов принятия решений, оптимизации бизнес-процессов и повышения эффективности работы компании.

Преимущества и недостатки использования нейронных сетей в системах поддержки принятия решений

Преимущества использования:

1. Обработка больших объемов данных.

Нейронные сети могут обрабатывать большое количество данных и выдавать точные прогнозы.

2. Улучшение качества принимаемых решений.

Нейронные сети позволяют учитывать различные факторы при принятии решений и выдавать более точные результаты.

3. Автоматизация процесса принятия решений.

Нейронные сети могут быть интегрированы в систему автоматического принятия решений, что ускоряет процесс и снижает вероятность ошибок.

Недостатки использования:

1. Сложность обучения и поддержания модели.

Нейронные сети требуют большого количества данных для обучения и настройки, а также специалистов для поддержания и оптимизации модели.

2. Необходимость высокой вычислительной мощности.

Для обработки больших объемов данных и обучения нейронных сетей требуется высокопроизводительное оборудование.

3. Необходимость точных данных.

Нейронные сети требуют точных данных для обучения и принятия решений, что может быть проблематично в некоторых областях.

В заключение можно отметить, что нейронные сети являются перспективной технологией для улучшения систем поддержки принятия решений. Они могут обрабатывать большие объемы данных и выдавать на выходе точный результат, основанный на анализе большого количества факторов.

Внедрение нейронных сетей в СППР позволяет повысить качество принимаемых решений и улучшить эффективность бизнеса в целом.

Однако необходимо учитывать ограничения этой технологии, такие как требование больших объемов данных для обучения и сложность интерпретации результатов.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Богданова Е.А. Применение нейронных сетей в системах поддержки принятия решений / Е.А. Богданова, О.А. Чернова, Е.А. Шаманова // Информационные технологии и математическое моделирование. – 2016. – Т. 23. – №. 2. – С. 89-96.

2. Жарков А.С. Нейронные сети в системах поддержки принятия решений: проблемы и перспективы / А.С. Жарков, О.В. Кузнецова, А.И. Назаренко // Информационные технологии и математическое моделирование. – 2018. – Т. 25. – №. 3. – С. 157-165.

3. Сазонова М.В. Применение нейронных сетей для прогнозирования рисков в системах поддержки принятия решений / М.В. Сазонова, Е.А. Лебедева, Е.Ю. Григорьева // Информатика и ее применения. – 2018. – Т. 12. – №. 2. – С. 42-49.

4. Шабалин В.В. Нейронные сети в системах поддержки принятия решений в условиях неопределенности/ В.В. Шабалин, Е.Ю. Григорьева, Е.А. Лебедева // Информационные технологии и вычислительные системы. – 2017. – Т. 5. – №. 4. – С. 27-33.

УДК 004.01

А.А. ПОГУДАЕВ
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

СПОСОБЫ МЕЖПРОЦЕССНОГО ВЗАИМОДЕЙСТВИЯ В ОС LINUX

В статье рассматриваются способы межпроцессного взаимодействия в ОС Linux, внимание акцентируется на сходстве и различии этих способов, а так же на удобстве их использования для программиста и безопасности.

В настоящее время существует много программ, которые выполняются на компьютере как несколько отдельных процессов. Такой подход используется по нескольким причинам:

1) Обеспечение параллельной обработки данных. На многопроцессорных системах это дает выигрыш в производительности.

2) Обеспечение стабильности. Каждый процесс имеет свою собственную память и при аварийном завершении одного процесса, остальные могут продолжить работу.

Однако разделение программы на несколько процессов приносит и определенные проблемы – процессам, как правило, необходимо как-то общаться между собой для координации действий и обмена данными.

В ОС Linux существует множество способов межпроцессного взаимодействия. Каждый из них имеет как свои преимущества, так и недостатки. При написании программ следует выбирать наиболее подходящий способ межпроцессного взаимодействия, так как иначе может снизиться производительность, или ухудшиться безопасность приложения.

Для начала следует понимать, как может быть организовано программное обеспечение, состоящее из нескольких процессов. Можно выделить несколько видов такого ПО.

1) Все процессы являются потомками главного процесса, который был запущен пользователем. Главный процесс через

некоторое время после запуска выполняет вызов fork и разделяется на два процесса: родительский и дочерний. После порождения нового процесса, программа выполняется дальше так, как будто вызова fork не было, и единственный способ узнать какой именно процесс сейчас выполняется - это проанализировать возврат функции fork. В дочернем процессе fork вернет значение 0, а в родительском – PID созданного дочернего процесса. Все процессы существуют строго в рамках одной вычислительной машины. Примером такого ПО могут быть web-сервера nginx и apache2.

2) Все процессы должны запускаться отдельно. В этом случае пользователь (или некоторый скрипт) запускает несколько отдельных программ, которые устанавливают каналы связи между собой и далее работают как единое целое. В отличии от первого случая тут отдельные процессы могут запускаться и на отдельных вычислительных машинах, которые связаны сетью. Примером могут служить ноды в ROS (Robot Operating System).

3) Смешанный подход. Некоторые процессы разделяются с помощью fork, а некоторые запускаются независимо.

Теперь рассмотрим способы взаимодействия программ.

Сигналы – один из самых древних способов межпроцессного взаимодействия. Первый процесс устанавливает функцию-обработчик определенного сигнала, а второй процесс посылает этот сигнал первому. Обработка сигнала первым процессом напоминает обработку прерывания – процесс перестает выполнять основной код и переходит к обработчику сигнала. После того, как функция-обработчик сигнала будет завершена, процесс продолжает выполнять основной код с того места, где он остановился. Момент получения сигнала непредсказуем, поэтому, как правило, функция-обработчик содержит минимальное количество кода, например установку какого-либо флага. Внутри обработчика можно вызывать только сигнал-безопасные функции. Обработчик сигнала устанавливается с помощью функции sigaction. А сигнал посылается с помощью функций raise, kill, killpg, pthread_kill, tgkill, sigqueue. С помощью sigqueue можно передать вместе с сигналом пользовательские данные, но их объем сильно ограничен. Это может быть переменная типа int или указатель на область памяти. При этом следует понимать, что передача указателя на область памяти не делает эту память автоматически доступной другому процессу. Существует возможность использовать сигналы в цикле обработки событий [1] с помощь функций ppoll и pselect. Также существует возможность остановить выполнение процесса до тех пор, пока не будет получен определенный сигнал (функция sigwait и её вариации).

Существует довольно много различных сигналов, но большинство из них зарезервировано под системные нужды. Для пользовательских нужд следует использовать сигналы SIGUSR1 и SIGUSR2. Для посылки сигнала необходимо знать PID процесса, которому этот сигнал отправляется. Если процессы, общающиеся посредством сигналов, не являются родителем и потомком, то узнать PID становится затруднительно, так как он присваивается операционной системой. У процесса получающего сигнал нет возможности узнать, кто является процессом отправителем, это сильно понижает безопасность, так как любой процесс может послать сигнал любому процессу, зная его PID.

Существует консольная утилита kill, которая позволяет послать любой сигнал любому процессу.

pipe – это однонаправленный канал передачи данных. Вызов функции pipe открывает два дескриптора, с которыми можно работать как с файловыми дескрипторами или с дескрипторами сокетов. Первый дескриптор может быть использован только для чтения, второй – только для записи. pipe создается в рамках одного процесса, после этого делается вызов fork и родительский и дочерний процесс могут общаться с помощью ранее созданного pipe. Например, родительский процесс будет писать данные во второй дескриптор, а дочерний считывать их из первого, или наоборот. pipe не дает гарантии, что данные будут считываться порциями такого же размера как были записаны. Однако данные не потеряются и не поменяют порядок. Это называется потоковой передачей данных. Минус потоковой передачи заключается в том, что потребуется предавать дополнительную информацию, чтобы понять, где конец одной посылки и начало другой. Так как у других процессов нет возможности записать или прочитать данные из pipe, то такой канал можно считать безопасным.

fifo – это тоже однонаправленный канал передачи данных. Очень похож на pipe, но имеет другой механизм создания. fifo создается с помощью функции mkfifo. Аргументом mkfifo является путь в файловой системе, по которому будет создан специальный объект, являющийся отображением этого канала. Затем необходимо открыть этот объект как обычный файл с обязательной передачей флага O_RDONLY или O_WRONLY для идентификации конкретной стороны канала. Функция open блокирует поток выполнения до тех пор, пока не будет открыт второй конец канала. После завершения использования fifo следует удалить объект в файловой системе, например с помощью функции unlink, иначе, он так и останется после

завершения работы программы. Часто unlink вызывают еще и перед созданием fifo для того, чтобы удалить старый fifo, если по каким-то причинам он не был удален. Существует так же консольная утилита mkfifo, которая делает то же самое, что и одноименная функция. Так как для использования fifo нужно только знать путь до объекта в файловой системе, то для его использования процессы могут запускаться независимо друг от друга. Передача данных идет в потоковом режиме. С точки зрения безопасности fifo достаточно надежен, так как только один процесс может читать, или писать данные в него одновременно.

Парные сокеты. Функция socketpair создает два связанных дескриптора сокета. Это похоже на pipe, но канал получается двунаправленным и доступна не только потоковая передача данных, но и пакетная. При пакетной передаче данных данные будут считываться точно такими же фрагментами, как были записаны. Первый аргумент функции socketpair всегда должен быть AF_UNIX, второй аргумент – режим, может принимать значения: SOCK_STREAM – потоковая передача данных; SOCK_DGRAM – пакетная передача данных (порядок и доставка не гарантируются); SOCK_SEQPACKET – пакетная передача данных (порядок и доставка гарантируется). Так как оба дескриптора создаются одновременно, то, как и в случае с pipe, socketpair должна вызываться до fork, а процессы должны быть родственными. При пакетной передаче данных размер пакета ограничен. Максимальный размер пакета на 32 байта меньше, чем опция сокета SO_SNDBUF. Значение SO_SNDBUF можно узнать и переопределить с помощью функций getsockopt и setsockopt.

Сетевые сокеты могут использоваться для взаимодействия процессов по сети в рамках классической клиент-серверной архитетектуры. Если взаимодействие требуется в пределах одной вычислительной машины, то в качестве адреса может использоваться localhost, при этом не требуется наличие физического сетевого адаптера. Сетевые сокеты создаются с помощью функции socket. Первый аргумент функции socket – это семейство протоков, он может принимать значения AF_INET (для IPv4) и AF_INET6 (для IPv6). Второй аргумент функции socket – это используемый режим (SOCK_STREAM, SOCK_DGRAM, SOCK_SEQPACKET). Логика при этом такая же, как и при использовании парных сокетов, но теперь каждый режим означает определенный сетевой протокол. SOCK_STREAM соответствует протоколу TCP, SOCK_DGRAM соответствует протоколу UDP, а SOCK_SEQPACKET соответствует довольно редкому протоколу SCTP. Следует заметить, что протокол

SCTP отсутствует на большинстве дистрибутивов linux. Для его использования потребуется пересобрать ядро или загрузить дополнительные модули ядра. Максимальный размер пакета при пакетной передаче также ограничен опцией сокета SO_SNDBUF, но помимо этого он не может быть больше, чем 65535 за вычетом суммы длин заголовков ip и udp/sctp. Например, для IPv4-UDP максимальный размер пакета (полезных данных) составит 65535-(20+8)=65507. С точки зрения API, SOCK_STREAM и SOCK_SEQPACKET схожи, имеют чётко выделенные роли клиента и сервера, и требуют предварительной установки соединения в отличие от сокетов SOCK_DGRAM, где роль клиента и сервера может быть размыта, а установка соединения не требуется. Сетевые сокеты могут быть удобны для отладки, так как позволяют перехватывать передаваемый трафик с помощью программ сниферов, таких как tcpdump, wireshark и других. Однако это же делает их более уязвимыми, зловредное ПО может генерировать пакеты-обманки и перехватывать пакеты с данными.

Сокеты домена UNIX или локальные сокеты. Такие сокеты, так же как и сетевые, создаются с помощью функции socket. Но в качестве семейства указывается константа AF_UNIX или AF_LOCAL (эти константы равны). Принцип работы полностью аналогичен сетевым сокетам. Но вместо сетевого адреса, при привязке сокета, задается путь в файловой системе. Как и в случае с pipe создается специальный объект в файловой системе, который следует удалить, когда он станет не нужен. Размер пакета при пакетной передаче ограничен, как и в случае с парными сокетами только опцией SO_SNDBUF. Такие сокеты не формируют сетевых пакетов, не требуют даже виртуального сетевого адаптера и данные через них не могут быть переданы за пределы одной вычислительной машины. По этой причине являются более безопасными и надежными чем сетевые сокеты. По API они почти идентичны сетевым сокетам, что делает возможным, при необходимости, быстро заменить в коде локальные сокеты на сетевые или наоборот.

Межпроцессное взаимодействие System V. System V IPC API существует в ОС Linux для совместимости с UNIX. Это API содержит функции для работы с очередями событий: msgget – создание или получение доступа к очереди; msgctl – управление очередью; msgsnd – отправка сообщений; msgrcv – прием сообщений. Функции для работы с семафорами: semget – создание или получение доступа к семафору; semop и semtimedop – блокировка и разблокировка семафора; semctl – получение информации и настройка семафора.

Функции для работы с общей памятью: shmget – создание блока общей памяти или получение доступа к ранее созданному блоку; shmat – привязывает общую память к текущему процессу; shmdt – отвязывает общую память от текущего процесса; shmctl – настройка общей памяти и получение информации о ней. Так же есть консольные утилиты ipcmk, ipcrm, ipcs, которые позволяют создавать/удалять объекты IPC и просматривать информацию о них. Для использования большинства этих функций программа должна обладать повышенными привилегиями, а именно CAP_IPC_OWNER. При отправке и приеме сообщений можно указать тип сообщения (переменная типа long), это позволяет использовать одну очередь большому количеству процессов, если использовать тип, как адрес назначения сообщения, то есть, каждый процесс будет читать из очереди сообщения только своего типа. На текущий момент System V IPC API считается устаревшим и не рекомендуется для новых разработок. Вместо него предлагается использовать POSIX IPC API.

Межпроцессное взаимодействие POSIX. POSIX IPC API пришло на замену устаревшему System V IPC API. В целом, POSIX IPC API имеет примерно такие же возможности, как и System V IPC API, но является более удобным и гибким. Функций стало больше, поэтому не буду приводить каждую из них. Функции делятся на те же классы: для работы с очередями событий; для работы с семафорами; для работы с общей памятью. В отличие от старого API в новом идентификаторы объектов являются файловыми дескрипторами, что позволяет отслеживать их состояние через функции poll/epoll/select. Повышенных привилегий не требуется. При посылке сообщения указывается приоритет. Сообщения с высоким приоритетом будут прочитаны из очереди раньше сообщений с низким приоритетом.

Следует понимать, что очереди, и System V и POSIX принципиально отличаются от сокетов, pipe и fifo, тем, что имеют один дескриптор и для чтения и для записи. То есть процесс может отправить сообщение в очередь и сам же его получить из того же самого дескриптора.

Передача больших объемов данных через общую память может быть очень эффективна и обгонять по производительности все другие способы, так как при передачи данных через сокеты, fifo, или pipe происходит копирование данных из памяти одного процесса в буфер в ядре, а затем из буфера ядра в память другого процесса, и именно копирование занимает большую часть времени. Однако, при использовании общей памяти легко допустить повреждение данных при одновременном доступе к ним из нескольких процессов.

Использование именованной общей памяти может быть не безопасно, так как злоумышленник может испортить данные в ней.

Все рассмотренные выше способы межпроцессного взаимодействия поддерживаются непосредственно ядром Linux, практически каждая функция API соответствует одноимённому системному вызову. Кроме этих способов существуют и более высокоуровневые, например шина D-Bus, или протокол MQTT и другие, но они являются всего лишь обертками над теми способами, что описаны в данной статье.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Погудаев А.А. Архитектура приложения, построенная на цикле обработки событий. // Новые информационные технологии в научных исследованиях: материалы XXVII Всероссийской научно-технической конференции студентов, молодых ученых и специалистов. – Рязань: ИП Коняхин А.В. (Book Jet), 2022. – С. 166.

УДК 004.65

Е.С. РАЙСКАЯ
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ СИСТЕМЫ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ ПОДБОРА АВТОМОБИЛЯ

Рассматриваются основные сущности для построения системы для предметной области для аренды автомобиля.

В данной работе основная цель программы – организация работы фирмы, занимающейся арендой автомобилей. Она имеет свой таксопарк, информация о котором хранится в базе данных. Также организация предоставляет необходимый набор документов, таких как акт приема-передачи транспортного средства, чек об оплате, договор аренды.

В общем виде входными параметрами является требования клиента к автомобилю и условиям аренды и информация о нем самом. Выходным же можно отнести договор и чек оп оплате аренды.

Для описания предметной области «Аренда автомобиля» на содержательном уровне необходимо выделить объекты, информация о

которых будет храниться в базе данных (сущности) и связи между ними.

В ходе работы для выбранной предметной области можно выделить следующие сущности и их первичные ключи:

1. Организация. Первичный ключ – код организации.
2. Автомобиль. Первичный ключ – идентификационный номер.
3. Автовладелец. Первичный ключ – *ID* владельца.
4. Клиент. Первичный ключ – *ID* клиента.
5. Страховка. Первичный ключ – номер полиса.
6. Город. Первичный ключ – код региона.
7. Ставка. Первичный ключ – код ставки.
8. Вид порчи. Первичный ключ – код порчи.
9. Акт приёма. Первичный ключ – номер акта.
10. Отчёт на доплату. Первичный ключ – номер отчёта.

Выделим и опишем связи [1].

1. Связь «Имеет». Данная связь между сущностями «Организация» и «Автомобиль» (рисунок 1). Тип связи – один ко многим. То есть у одной организации может быть много автомобилей. Следовательно, кардинальность «Организация – Автомобиль» составляет 0,n.

Рисунок 1 – Связь «Имеет»

2. Связь «Сдаёт». Данная связь между сущностями «Автовладелец» и «Автомобиль» (рисунок 2). Тип связи – один ко многим. То есть один автовладелец может сдавать несколько авто, но один автомобиль может иметь только одного владельца. Следовательно, кардинальность «Автовладелец – Автомобиль» составляет 0,n.

Рисунок 2 – Связь «Сдаёт»

3. Связь «Имеет». Данная связь между сущностями «Автомобиль» и «Страховка» (рисунок 3). Тип связи – один ко многим. То есть у одного автомобиля может быть несколько страховок, но у одной страховки может быть только один автомобиль. Следовательно, кардинальность «Автомобиль – Страховка» составляет 0,n.

Рисунок 3 – Связь «Имеет»

4. Связь «Договор аренды». Данная связь между сущностями «Клиент» и «Автомобиль» (рисунок 4). Тип связи – многие ко многим. То есть один клиент может оформить аренду на разные автомобили, и один автомобиль может арендоваться разными клиентами. Следовательно, кардинальность «Клиент – Автомобиль» составляет m,n. Далее с помощью этой связи будут составляться договоры аренды.

Рисунок 4 – Связь «Договор аренды»

Эту связь можно заменить на 2 связи типа один ко многим – это «Договор аренды – Автомобиль» (рисунок 5) и «Клиент – Договор аренды» (рисунок 6).

Рисунок 5 – Связь «Договор аренды – Автомобиль»

Рисунок 6 – Связь «Клиент – Договор аренды»

5. Связь «Вид порчи по акту». Данная связь между сущностями «Вид порчи» и «Акт приёма» (рисунок 7). Тип связи – многие ко многим. То есть в одном акте приёма может быть несколько видов порчи, а один вид порчи может фигурировать в нескольких актах приёма. Следовательно, кардинальность «Вид порчи – Акт приёма» составляет m, n. Далее с помощью этой связи будут составляться отчёты по вид порчи по акту.

Рисунок 7 – Связь «Вид порчи по акту»

Эту связь можно представить в виде двух связей типа один ко многим – «Вид порчи по акту – Акт приёма» (рисунок 8) и «Вид порчи – Вид порчи по акту» (рисунок 9).

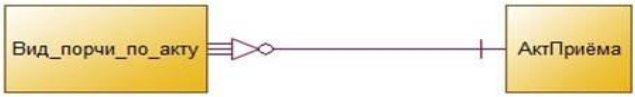


Рисунок 8 – Связь «Вид порчи по акту – Акт приёма»

Рисунок 9 – Связь «Вид порчи – Вид порчи по акту»

6. Связь «Отчёт». Данная связь между сущностями «Акт приёма» и «Отчёт на доплату» (рисунок 10). Тип связи – один ко многим. То есть у одного акта приёма может быть несколько отчётов на доплату, но у одного отчёта на доплату может быть только один акт приёма. Следовательно, кардинальность «Акт приёма – Отчёт на доплату» составляет 0, n.

Рисунок 10 – Связь «Отчёт»

7. Связь «Арендуется в». Данная связь между сущностями «Город» и «Договор аренды» (рисунок 11). Тип связи – один ко многим. То есть в одном городе может быть несколько аренд, но одна аренда может происходить только в одном городе. Следовательно, кардинальность «Город – Договор аренды» составляет 0, n.

Рисунок 11 – Связь «Арендуется в».

8. Связь арендуется по». Эта связь между сущностями «Ставка» и «Договор аренды» (рисунок 12). Тип связи – один ко многим. То есть одна ставка может быть указана в нескольких договорах аренды, но в одном договоре аренды может быть использована только одна ставка. Следовательно, кардинальность «Ставка – Договор аренды» равна 0, n.

Рисунок 12 – Связь «Арендуется по»

9. Связь «Оформляется». Данная связь между сущностями «Договор аренды» и «Акт приёма-передачи» (рисунок 13). Тип связи – один ко многим. То есть на один договор аренды можно составить несколько актов приёма-передачи, но один акт приёма-передачи может использоваться только в одном договоре аренды. Следовательно, кардинальность «Договор аренды – Акт приёма-передачи» равна 0, n.

Рисунок 13 – Связь «Оформляется»

В результате разработки получаем инфологическую модель, представленную на рисунке 14.

Переход к логическому моделированию происходит на основе разработанной инфологической модели. На этом этапе сущности дополняются остальными атрибутами, а связи многие ко многим заменяются дополнительной таблицей и двумя связями один ко многим. В нашем случае появились таблицы «Договор аренды» и «Вид порчи по акту».

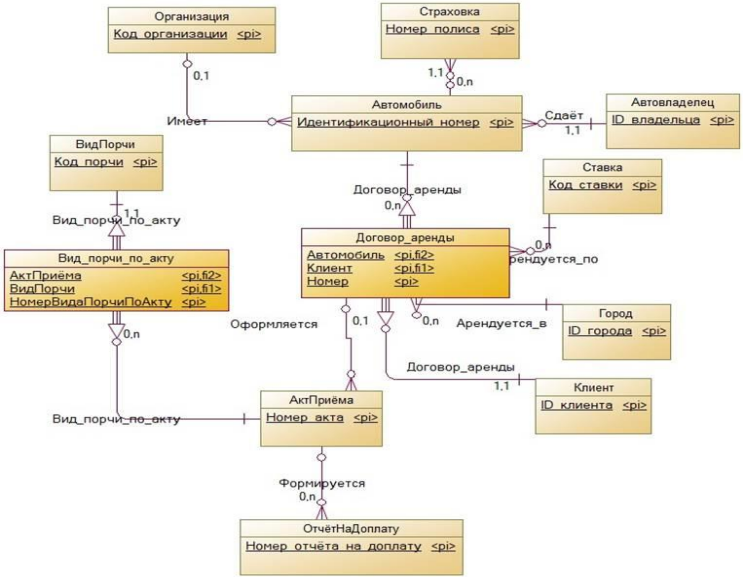


Рисунок 14 – Инфологическая модель

Выделим необходимые атрибуты для всех сущностей.

Атрибуты сущности «Организация»:

- код организации;

- название;
- ИНН;
- ОГРН;
- дата регистрации;
- телефон;
- ФИО директора;
- адрес.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Автоматизация проектирования баз данных в среде Sybase PowerDesigner: методические указания к лабораторным работам / Рязан. гос. радиотехн. Ун-т.; сост.: А.В. Благодаров, Р.В. Тишкин. – Рязань, 2010. – 37 с.

УДК 621.371.391.1

С.С. РУМЯНЦЕВ
АО «Моринформсистема-Агат-КИП»

ИЗМЕРЕНИЕ УРОВНЯ ВОДЫ В КОНТУРЕ ЯДЕРНОЙ ЭНЕРГЕТИЧЕСКОЙ УСТАНОВКИ РЕФЛЕКТОМЕТРИЧЕСКИМ МЕТОДОМ

В статье исследован импульсный рефлектометрический уровнемер применительно к измерению уровня теплоносителя в ядерной энергетической установке. Построена математическая модель рефлектометра и проведен расчёт рефлектрограммы. Сделаны рекомендации по повышению точности измерений.

Предупреждение возникновения аварийных ситуаций и надёжное и безопасное функционирование ядерных энергетических установок требует контроля параметров в тепловых контурах, в том числе в «грязном» контуре, теплоноситель которого непосредственно омывает активную зону. Одной из задач, решаемых в рамках данной проблемы, является необходимость измерения уровня жидкости-теплоносителя, в качестве которого, как правило, выступает дистиллированная вода. Датчики-уровнемеры в подобных системах работают в экстремальных условиях, подвергаясь жёсткому и интенсивному радиоактивному облучению, воздействию температуры до 250 °С и давлению до 25 МПа.

Это накладывает ряд дополнительных требований, в том числе необходимость достаточной механической прочности конструкции уровнемера, герметчиность, в интересах недопущения утечек из

контура высокого давления, кроме того, невозможно разместить электрическую часть рядом с активной зоной реактора.

Способом решения данной задачи является использование импульсных рефлектометров, применяемых для измерения уровня жидкости и сыпучих материалов [1]. К достоинствам данного технического решения следует отнести: возможность вынести приёмопередатчик с устройством обработки за пределы «грязной» зоны, технологичность конструкции датчика чувствительного элемента и возможно обеспечить герметичность контура ядерной энергетической установки в случае отказа или частичного разрушения чувствительного элемента.

Расчётная математическая модель СВЧ-тракта импульсного рефлектометра показана на рисунке 1.

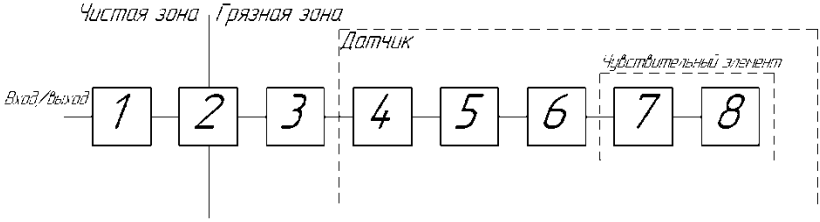


Рисунок 1 – Модель СВЧ-тракта импульсного рефлектометра (1, 3 – коаксиальный кабель; 2 – переходная муфта; 4, 6 – изолятор; 5 – межизоляционный промежуток; 7 и 8 – незатопленная и затопленная части чувствительного элемента)

Формирование зондирующего сигнала происходит в приёмопередатчике, который соединяется коаксиальным кабелем с переходной муфтой, вмурованной в стену, отделяющую «чистую» зону от «грязной». В «грязном» помещении датчик, состоящий из двух изоляторов с межизоляционным промежутком и чувствительного элемента, соединяется с переходной муфтой коаксиальным кабелем длиной до 30 метров.

Каждый элемент СВЧ-тракта, представленной на рисунке 1, можно описать в виде отрезка линии с отражающей неоднородностью на конце (рисунок 2).

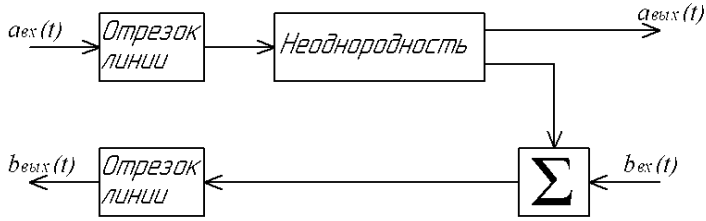


Рисунок 2 – Модель отрезка линии с неоднородностью на конце

При этом *i-й* отрезок импульсного рефлектометра характеризуется входной $a_{вхi}$ и выходной $a_{выхi}$ падающими и входной $b_{вхi}$ и выходной $b_{выхi}$ отраженными волнами, которые связаны выражениями:

$$\begin{cases} a_{выхi}(t) = V_{Лi} T a_{вхi}(t-\tau_i) \\ b_{выхi}(t) = V_{Лi}{}^2 R_i a_{вхi}(t-2\tau_i) + V_{Лi} T_i b_{вхi}(t-\tau_i) \end{cases}, \quad (1)$$

где $V_{Лi}$ – коэффициент передачи *i-го* участка импульсного рефлектометра; T_i – коэффициент прохождения неоднородности на границе раздела *i-го и i+1-го* участка импульсного рефлектометра; R_i – коэффициент отражения неоднородности на границе раздела *i-го* и *i+1-го* участка импульсного рефлектометра; τ_i – задержка распространения сигнала в *i-м* участке импульсного рефлектометра, с.

Задержка распространения сигнала в *i-м* участке импульсного рефлектометра определяется по формуле:

$$\tau_i = l_{лi} \Big/ v_{лi}, \quad (2)$$

где $l_{лi}$ – длина *i-го* участка импульсного рефлектометра, м; $v_л$ – скорость распространения сигнала на *i-м* участке импульсного рефлектометра, м/с.

При этом скорость уменьшается, пропорционально коэффициенту замедления $1/\sqrt{\varepsilon}$, где ε – относительная диэлектрическая проницаемость [2].

Следует отметить, что диэлектрическая проницаемость воды и сухого пара зависит от температуры и давления, которые необходимо оперативно измерять, в интересах калибровки чувствительного элемента. Закономерности изменения диэлектрической проницаемости насыщенного водяного пара приведены в работе [3].

Коэффициент передачи *i-го* участка СВЧ-тракта импульсного рефлектометра $V_{Лi}$ определяется по формуле:

$$V_{Лi} = 10^{-\frac{\alpha_{Лi} l_{Лi}}{20}}, \quad (3)$$

где $\alpha_{Лi}$ – коэффициент затухания, дБ/м.

Коэффициенты отражения R_i и прохождения T_i определяются по формулам Френеля [2]:

$$R_i = \frac{Z_{i+1}\cos\varphi - Z_i \cos\psi}{Z_{i+1}\cos\varphi + Z_i \cos\psi}, \quad (4)$$

где Z_i и Z_{i+1} – волновые сопротивления *i-го* и *i+1-го* участков соответственно, преобразуется к виду:

$$R_i = \frac{Z_{i+1} - Z_i}{Z_{i+1} + Z_i}, \quad (5)$$

а коэффициент прохождения *i-го* участка импульсного рефлектометра

$$T_i = 1 - |R_i|. \quad (6)$$

Моделирование рефлектограммы проведем по формуле (1), выбрав следующие параметры: волновые сопротивления соединительных кабелей возьмем равными 75 Ом (соответствует применяемому в подобных системах кабелю РК-75-7-22). Отклонение волнового сопротивления Изоляторов и переходной муфты примем равным ±10%. Параметры для каждого участка СВЧ тракта импульсного рефлектометра сведём в таблицу 1.

Таблица 1 – Параметры участков СВЧ тракта импульсного рефлектометра

№	Элемент	Z_i, Ом	$l_{лi}$, м	ε	$\alpha_{Лi}$, дБ/м
1	Соединительный кабель	75	1	2	0,33
2	Переходная муфта на основе боросиликатного стекла	67,5…82,5	0,25	6,2	0,37
3	Соединительный кабель	75	30	2	0,33
4	Изолятор-1	67,5…82,5	0,02	6,2	0,37
5	Промежуток между изоляторами	75	0,3	1	0,027
6	Изолятор-2	67,5…82,5	0,02	6,2	0,37
7	Незатопленная часть чувствительного элемента	75	5	1	0,027
8	Затопленная часть чувствительного элемента	8,3	1	81	0,33

Рефлектограмма импульсного рефлектометра для исходных данных показанных в таблице 1 приведена на рисунке 3.

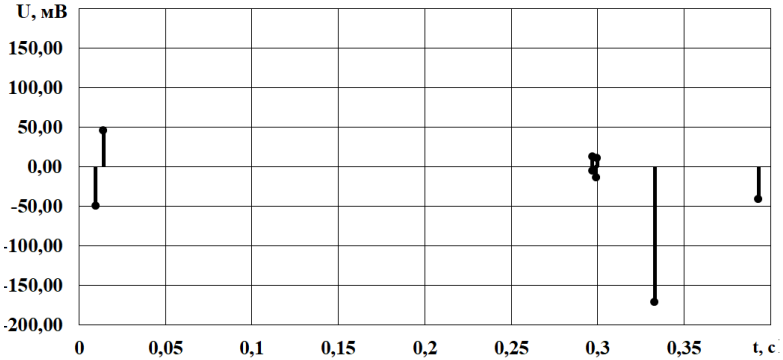


Рисунок 3 – Рефлектограмма импульсного рефлектометра для случая затопления чувствительного элемента датчика на 1 м

Учитывая достаточно большую амплитуду эхо-сигнала отраженного от конца чувствительного элемента, которая при данных условиях составляет 40 мВ (амплитуда зондирующего сигнала на входе приёмника 1 В), Оценку уровня воды можно выполнить используя, как эхо-сигнал от границы раздела сред, так и эхо сигнал от конца чувствительного элемента, амплитуда которого при данных условиях составляет 40 мВ (амплитуда зондирующего сигнала на входе приёмника 1 В). Величина временной задержки импульса в затопленной и незатопленной части, в зависимости от длины затопленной части чувствительного элемента показана на рисунке 4.

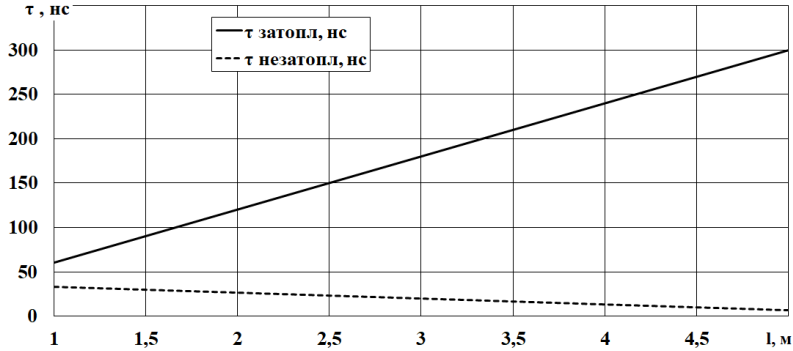


Рисунок 4 – Величина временной задержки импульса в затопленной и незатопленной частях чувствительного элемента импульсного рефлектометра

При этом амплитуда эхосигнала от конца чувствительного элемента составляет 30-40 мкВ, а амплитуда эхо-сигнала от границы раздела сред 170,5-175 мкВ. В таких условиях может быть целесообразнее измерять длительность эхо-сигнала от конца чувствительного элемента, т.к. её возможно точнее измерить вследствие большей длительности, в то время как непосредственно наличие отраженного импульса достаточно обнаружить.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Тренкаль Е.И. Измерение уровней жидкости методом импульсной рефлектометрии (обзор) / Е.И. Тренкаль, А.Г. Лощилов // Доклады ТУСУР. – 2016. – Т. 19, № 4. – С. 67-73.

2. Баскаков С.И. Основы электродинамики / С.И. Баскаков Учебное пособие для вузов. – М., «Сов. радио». – 1973. – 248 с.

3. Fernandez D.P. A Formulation for the Static Permittivity of Water and Steam at Temperature from 238 K to 873 K at Pressures up to 1200 MPa, Including Derivatives and Debye-Hukel Coefficients / D.P. Fernandez, A.R.H. Goodwin, E.W. Lemmon et al. // J. Phys. Chem. Ref. Data. – 1997. – Vol. 26. – Pp. 1125-1166.

УДК 004.023

А.Н. САПРЫКИН, А.О. САПРЫКИНА
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ЭТАПЫ МОДИФИЦИРОВАННОГО ГЕНЕТИЧЕСКОГО АЛГОРИТМА ЗАДАЧИ КОМПОНОВКИ КОНСТРУКТИВНЫХ МОДУЛЕЙ ЭЛЕКТРОННЫХ СРЕДСТВ

Рассматриваются особенности разработанного авторами модифицированного генетического алгоритма. Описываются основные этапы его работы, детально рассматривается процесс формирования популяций и отбора потомков, расчета значения соответствующего значения полезности.

В классическом генетическом алгоритме новая популяция создается из потомков текущего поколения [1]. Однако для решения некоторых задач более эффективным становится применение модифицированного генетического алгоритма, для которого нет необходимости в создании новой популяции – можно использовать одну единственную, добавляя в нее новых особей и исключая «ненужных».

Решением задачи компоновки является информация, в которой указано разбитие элементов по блокам. Авторами уже исследовались возможности модифицированного генетического алгоритма [2]. Функция полезности особи представляет собой сумму значений полезности каждого блока.

В данной работе был использован один критерий для оценки полезности, а именно количество выходящих из блока цепей. Чем это значение меньше, тем выше полезность, таким образом, для блока она обратно пропорциональна количеству выходящих из него цепей. Для того чтобы рассчитать значение функции полезности, необходимо обращение к матрице инцидентности, содержащей в себе информацию обо всех соединениях элементов.

Схема модифицированного генетического алгоритма для решения задачи компоновки представлена на рисунке 1. Рассмотрим подробнее этапы работы представленного модифицированного генетического алгоритма.

Создание начальной популяции

Каждому элементу в хромосоме особи (структура «блоки по элементам») назначается номер случайного блока, который на момент начала работы еще не заполнен полностью. Одновременно с этим структура «элементы по блокам» заполняется номерами элементов. После заполнения структур автоматически запускается процедура подсчета полезности данной особи.

Таким образом создается новая особь. Их требуется ровно такое количество, чтобы размер популяции был равен заданному (исходному) значению размера популяции. При создании новых особей не должно происходить переполнение блоков, в которые входят генерируемые элементы, или повторы их номеров.

Вычисление полезности каждой особи

Функция полезности каждой отдельно взятой особи рассчитывается путем суммирования значений полезности каждого входящего в нее блока. Оценка полезности особи проводится критериально, по количеству выходящих из блока цепей. Особь считается обладающей более высоким значением полезности при меньшем значении данного параметра.

Выбор очередной пары родителей

Данный этап вызывается в отдельной процедуре по скрещиванию особей. Чем больше генерируется потомков, тем больше шанс, что будет найдено решение (особь) лучше, поэтому для скрещивания рассматриваются все возможные пары особей. Модифицированный генетический алгоритм может контролировать

количество скрещиваемых особей при помощи вероятности скрещивания (является исходными данными), которая применяется к каждой рассматриваемой паре. Если пара особей проходит «конкурс», то скрещивание происходит и в популяцию заносится новая особь – их потомок.

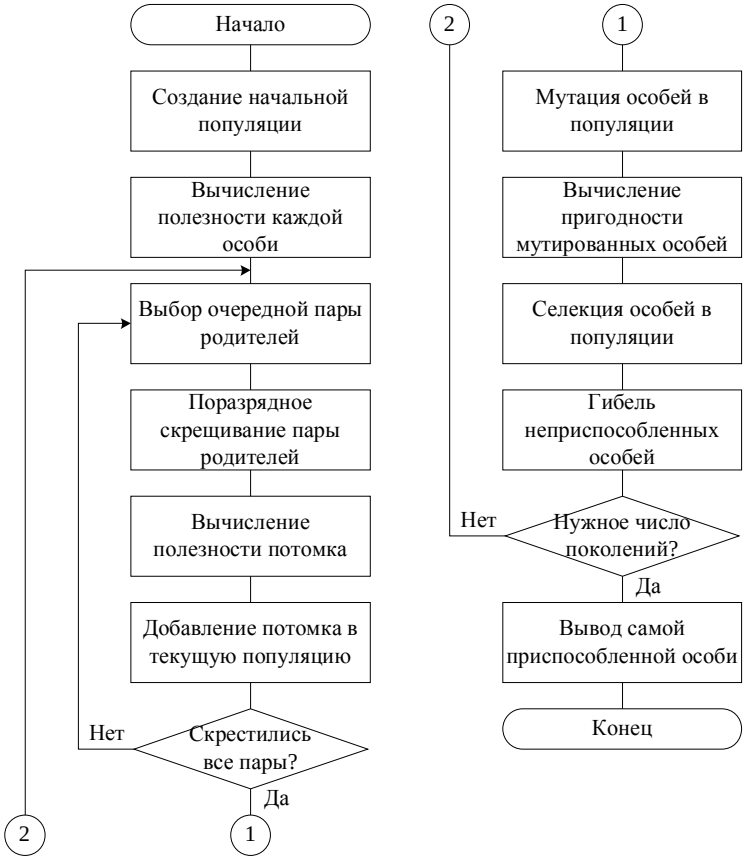


Рисунок 1 – Схема модифицированного генетического алгоритма

Скрещивание пары родителей

На данном этапе происходит процесс непосредственного скрещивания выбранной пары особей (родителей), в результате чего получается новая особь, которая будет содержать в себе гены обоих родителей. Как только алгоритм решил, что данная пара особей будет скрещиваться, создается новая особь. Каждый ген новой особи

заполняется значением гена той же позиции одного из родителей. Если значение гена родителя содержит номер блока, который уже заполнен, то выбирается ген другого родителя. Если же заполнены блоки генов обоих родителей, то в таком случае выбирается случайный свободный блок. Если родители были одинаковыми, то потомок будет таким же.

Для того чтобы не было случаев переполнения блоков, необходимо ввести систему контроля и управления такими случаями. Для решения данной задачи рассматриваемый модифицированный генетический алгоритм использует поразрядное или многоточечное скрещивание с генерацией лишь одного потомка.

Вычисление полезности потомка

После создания новой особи необходимо рассчитать ее полезность, чтобы определить ее ценность.

Добавление потомка в текущую популяцию

После вычисления полезности новой особи, алгоритм добавляет ее в текущую популяцию.

Все родители участвовали в скрещивании?

Процесс скрещивания продолжается до тех пор, пока все пары особей не попробуют между собой скреститься. После того, как процесс скрещивания всех пар особей завершается, необходимо определить какая особь является лучшей по параметру полезности.

Мутация особей в популяции

Каждая особь в популяции проходит «конкурс», в результате чего особь либо будет подвергаться операции мутации, либо останется без изменений. Процесс мутации заключается в изменении случайного гена. Если произошло переполнение блока, то в нем выбирается случайный элемент (кроме текущего), и он переходит в предыдущий блок.

Вычисление пригодности мутированных особей

Если особь была мутирована, то необходимо обновить информацию о ее полезности. По завершению процесса мутации, если хотя бы одна особь была мутирована, то необходимо снова произвести поиск лучшую особь в популяции.

Селекция особей в популяции

На данном этапе происходит отбор наиболее полезных особей одним из предоставляемых методов: турнирный, усечением и метод рулетки. Метод усечением подразумевает быструю сортировку. По завершении любого из методов селекции происходит автоматическая гибель неприспособленных (и, как следствие, неотобранных) особей.

Гибель не приспособленных особей

Все особи в популяции, которые не были отобраны селекцией, исключаются из популяции.

Достигнуто предельное число поколений?

Если количество поколений не достигло желаемого количества, то вышеописанные процессы продолжаются с оставшимися после отбора особями.

Вывод самой приспособленной особи

В итоге выводится решение, которое наиболее эффективно по сравнению с другими в популяции.

Таким образом, авторами предложен модифицированный генетический алгоритм компоновки конструктивных модулей электронных средств с видоизмененными генетическими операторами. Подробно описаны основные этапы его работы.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Гладков, Л.А. Генетические алгоритмы: учебник / Л.А. Гладков, В.В. Курейчик, В.М. Курейчик; под ред. В.М. Курейчик. – Москва: Физматлит, 2010. – 317 с.

2. Сапрыкина, А.О. Использование генетического алгоритма эволюционной оптимизации в задаче компоновки конструктивных модулей электронных средств / А.О. Сапрыкина, А.Н. Сапрыкин // Материалы VII Всероссийской научно-технической конференции «Актуальные проблемы современной науки и производства». – Рязань: РГРТУ, 2022. – С. 270-275.

УДК 004.9

А.Н. САПРЫКИН, А.А. ХРАМОВА, Т.С. ВАСИЛЬЕВА
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

РАЗРАБОТКА КОМПОНЕНТОВ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА ВЕБ-ОРИЕНТИРОВАННОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ

Рассматриваются компоненты пользовательского интерфейса, а также их разработка при проектировании интерфейса веб-ориентированной информационной системы.

В настоящее время существует большое количество веб-ориентированных информационных систем (ИС) с разнообразными пользовательскими интерфейсами. Пользовательский интерфейс – это средство взаимодействия пользователя (клиента) с программной частью ИС. Он позволяет упростить взаимодействие со сложными техническими объектами.

Пользовательский интерфейс состоит из компонентов, каждый из которых имеет свое назначение. Рассмотрим основные из них.

Элемент «Header»

Начнем с элемента, расположенного в верхней части экрана, называемого хедер или «шапка». Header является областью захвата внимания клиентов. Он служит для осуществления навигации по ИС.

Основными элементами Header являются:

- логотип организации;
- контактная информация;
- горизонтальное верхнее меню.

Также Header может включать в себя следующие элементы:

- гамбургер-меню;
- форму поиска;
- ссылки на аккаунты организации в социальных сетях;
- форму обратной связи.

На рисунке 1 представлен пример элемента «Header».

Рисунок 8 – Пример компонента «Header»

Слайдер

Трудно представить себе современную веб-ориентированную ИС, не содержащую слайдер. Визуально он является блоком на странице, в пределах которого с установленной периодичностью происходит демонстрация предпросмотров изображений, новостей или статей. С помощью него можно красочно и кратко представить данные для привлечения пользователей. Также слайдер позволяет сэкономить пространство на странице, т.к. на одном блоке можно разместить сразу несколько информационных сообщений.

Слайдеры отличаются друг от друга различным функционалом:

- режим последовательного просмотра;
- просмотр с помощью свайпов;
- просмотр с помощью элементов управления;
- гибридный просмотр (радиокнопки, боковые стрелки).

Зачастую слайдер создается с помощью таких языков программирования, как HTML, CSS и JavaScript, но также существует множество готовых решений (библиотека Swiperjs и т.д.). На рисунке 2 представлен пример слайдера.

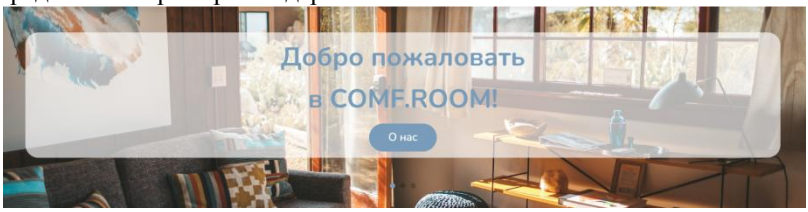


Рисунок 9 – Пример компонента «слайдер»

Блок с контентом

Контентная часть или work area, часто используется при верстке пользовательского интерфейса. Контентом называют информацию, представленную в веб-ориентированной ИС в различной форме. Это тексты, изображения, а также информация в аудио- и видео форматах.

Благодаря использованию контентных блоков можно уникально оформить текстовые данные и картинки, как угодно расположив их на странице. На рисунке 3 представлен пример блоков с контентом.

Рисунок 10 – Пример компонента «блок с контентом»

Кнопка

Кнопки являются одним из базовых элементов любого пользовательского интерфейса. Они играют важную роль в организации взаимодействия между пользователем и интерфейсом.

Кнопка является наиболее известным интерактивным объектом, который может обрабатывать любое событие в зависимости от поставленной задачи, ведь её основная функция – управление. Кнопки эффективно имитируют взаимодействие с объектами в материальном мире. Так, если кликнуть по кнопке входа, произойдет вход в аккаунт

пользователя, а кнопка быстрого просмотра товара переведет клиента на страницу с подробной информацией и продукте и т.д.

Современные кнопки пользовательского интерфейса разнообразны и многофункциональны. Как правило, они представляют собой интерактивную зону, которая заметно выделяется среди остальных объектов, имеет определенную геометрическую форму и поддерживается текстом, объясняющим выполняемое действие.

Рассмотрим основные виды кнопок для пользовательского интерфейса:

- кнопка призыва к действию – побуждает клиентов к выполнению тех или иных действий, превращая их из пассивных пользователей в активных;
- текстовая кнопка – кнопка с определенным текстом (часто используется для навигации);
- кнопка с выпадающим списком – при нажатии на нее отображается выпадающий список взаимоисключающих пунктов;
- кнопка «гамбургер» – с помощью нее открывается развернутый список меню;
- кнопка с выбором вариантов – при нажатии на данную кнопку открывается набор доступных функций;
- кнопка «поделиться» – упрощает взаимодействие ИС с аккаунтом пользователя в социальной сети, для быстрой передачи данных;
- плавающая кнопка действия – представляет собой кнопку, парящую над контентом страницы. Обеспечивает мгновенный доступ к наиболее важным и популярным действиям пользователя на экране приложения.

Форма и ее поля

Отметим компонент для взаимодействия пользователя с веб-страницей, называемый веб-формой. Она является специально ограниченной областью интерфейса. Пользователь может внести в нее необходимую информацию, а также выбрать конкретные действия из предложенных. Веб-форма является аналогом бумажной формы, различных анкет.

Формы состоят из различных полей для ввода данных. Рассмотрим основные виды полей ввода:

- поле для ввода текста (type = «text») – в данное поле можно вводить любой тип данных;
- поле для ввода даты (type = «date») – определено для задания даты;

- поле для ввода электронной почты (type = «email») – поле для проверки корректности ввода E-mail;
- поле для ввода чисел (type = «number») – предназначено только для ввода чисел (ввести буквы или другие символы не получится);
- поле для ввода пароля (type = «password») – в данное поле можно вводить различные символы клавиатуры, но вместо них будут отображаться точки;
- поле для ввода номера телефона (type = «tel») – позволяет вводить только цифры, в случае использования интерфейса с мобильного устройства переключает клавиатуру на ввод чисел;
- поле для ввода в виде кнопки (type = «range») – устанавливает числовое значение в пределах заданных минимума и максимума;
- поле для ввода в виде кнопки (type = «submit») – при нажатии отправляет данные на сервер, если он расположен внутри формы. На рисунке 4 представлен пример формы с полями ввода.

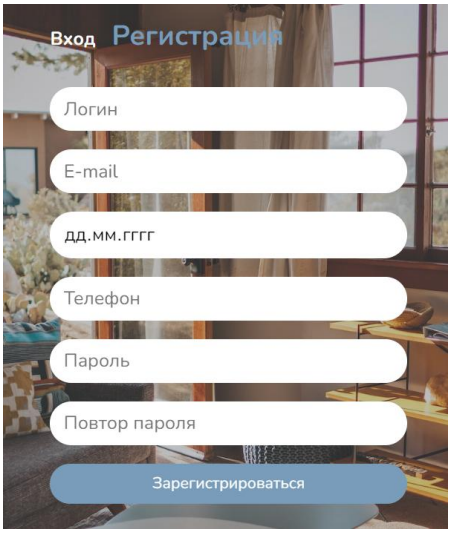


Рисунок 11 – Пример компонента «форма»

Выпадающий список

Одним из способов организации материалов на веб-странице является выпадающий список, позволяющий создать список с подпунктами (при желании). При выполнении таких действий, как нажатие или наведение мышью, компонент раскрывается. Он удобен

для сокрытия небольшого количества вариантов выбора для пользователя. На рисунке 5 представлен пример выпадающего списка.

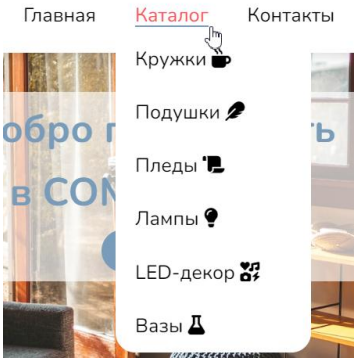


Рисунок 12 – Пример компонента «выпадающий список»

Аккордеон

Аккордеон – это вертикально сложенный набор заголовков. Каждый из них может быть развернут, чтобы показать содержимое, связанное с ним. Этот элемент так называется из-за того, что принцип его действия напоминает музыкальный инструмент аккордеон.

Существует несколько вариантов использования аккордеона:

- для разбиения информации на части и уменьшения беспорядка в ее размещении;
- использование аккордеона в качестве навигации для группировки ссылок.

Всплывающее окно

Всплывающее окно или Pop Up – это окно, которое появляется поверх страницы. Зачастую оно полностью забирает внимание: пока пользователь не закроет его, он не сможет просматривать страницу.

Всплывающее окно может использоваться для анонса новинок, изменений в компании, для рассказа об акциях и скидках. Клиентам не нужно искать информацию на сайте: они могут сразу ознакомиться с предложением и оставить контакты для связи с менеджером. Также всплывающие окна используются для сбора контактных данных.

Преимущества всплывающих окон:

- можно выводить в конкретном положении при скролле;
- возможно сразу отображать окно, либо с временной отсрочкой;
- отображать после действия пользователя.

Рассмотрим основные типы всплывающих окон:

- лайтбокс – располагается на переднем плане, при этом затемняя задний фон;
- «ДА»/«НЕТ» – используется для проведения односложных опросов в несколько шагов;
- геймифицированное – всплывающее окно в виде простой игры, выигрышем в которой может быть скидка или другой бонус;
- небольшой боковой баннер – не мешает просматривать контент, хорошая альтернатива для рекламы.
- полноэкранное – всплывающее окно, которое занимает весь экран интерфейса и полностью забирает внимание пользователя (зачастую со стороны клиента воспринимается с агрессией).

Строка поиска

Поисковая строка помогает найти нужную информацию на странице либо с помощью внутренних инструментов сайта, либо благодаря готовым решениям от Яндекса или Google.

Медиаплеер

Медиаплееры – это решения, которые позволяют воспроизводить медиаконтент. Показ аудио/видео делает интерфейс более информативным и легче воспринимается пользователем. Медиаплеер можно создать с помощью таких языков программирования, как HTML и JavaScript, или можно воспользоваться веб-плеерами с открытым исходным кодом: Video.js, Shaka Player, Plyr, jPlayer и т.д.

Элемент «Footer»

Футер или подвал сайта расположен в нижней части экрана. Функции футера и хедера схожи. Обычно он включает такие компоненты, как контактная информация организации, карта, указывающая расположение офиса, а также ссылки на социальные сети. В футере возможно дублирование разделов сайта. Обязательными элементами футера является: политика конфиденциальности, авторское право и условия использования.

С помощью футера можно взаимодействовать с клиентом в тот момент, когда он пролистал страницу до конца. С помощью контента в футере можно побудить клиента к переходу на другие страницы или оставить свои контактные данные (обратная связь).

Компоненты пользовательского интерфейса веб-ориентированной ИС позволяют правильно структурировать контент на странице, привлекают внимание пользователей на особо важную информацию, а также используются для дополнения дизайна интерфейса.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Спицина, И.А. Разработка информационных систем. Пользовательский интерфейс: учебное пособие для СПО / И.А. Спицина, К.А. Аксёнов; под редакцией Л.Г. Доросинского. – 2-е изд. – Саратов, Екатеринбург: Профобразование, Уральский федеральный университет, 2020. – 98 с.

2. Малышев, К.В. Построение пользовательских интерфейсов / К.В. Малышев. – Москва: ДМК Пресс, 2021. – 268 с.

3. Никулова, Г.А. Проектирование и реализация Web-интерфейса: учебно-методическое пособие / Г.А. Никулова.– Липецк: Липецкий государственный педагогический университет имени П.П. Семёнова-Тян-Шанского, 2020. – 63 с.

УДК 004.9

А.О. САПРЫКИНА
Рязанский государственный университет имени С.А. Есенина,
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ЭЛЕКТРОННОЕ ПОРТФОЛИО И ВИДЫ КОНТРОЛЯ РЕЗУЛЬТАТОВ ОБУЧЕНИЯ

В статье описываются возможности по применению технологии электронного портфолио в образовательном процессе. Анализируется его применение при осуществлении контроля за образовательными достижениями обучающихся, особенности применения при каждом виде контроля.

В наши дни возможности по применению информационных технологий в сфере образования только расширяются. Развитие онлайн-технологий и электронных образовательных ресурсов (ЭОР) открыло множество новых возможностей не только для улучшения учебного процесса в высшем образовании и упрощения процесса трудоустройства [1], но и для реформирования системы оценки образовательных результатов обучающихся.

Традиционно эта система представлена «оценками», которые количественно оценивают знания и навыки по изучаемому предмету. Зачастую данные оценки формальны и не всегда позволяют правильно оценить реальный уровень знаний или индивидуальный ответ каждого обучающегося. Технология портфолио позволяет разделить оценки на качественные и количественные и рассчитать на их основе комбинированную оценку результатов обучения.

Электронное портфолио представляет собой коллекцию учебных объектов и документов, представленных в электронном виде в каком-либо сервисе, будь то мобильное приложение, облачный сервис или целая социальная сеть, объединяющая вышеперечисленное. Электронное портфолио можно также использовать в качестве инструмента коммуникации и взаимодействия [2]. В облаке каждый пользователь создает свое личное пространство, в котором могут быть собраны файлы различных форматов, например, сертификаты, свидетельства о достижениях, презентации, результаты экзаменов и самостоятельные творческие работы. Уникальность электронного портфолио как образовательной технологии заключается в том, что оно позволяет обучающимся не только выполнять свою работу, но и демонстрировать ее и обсуждать с другими обучающимися и педагогами, что может стать источником сильной мотивации. Портфолио также известно как образовательный инструмент для самокритики и самоконтроля [2].

Контроль результатов обучения является неотъемлемой частью образовательного процесса. К нему предъявляется множество требований, таких как индивидуальный подход, систематичность, разнообразие форм, полнота, объективность, различные подходы к оценке разных предметов и единство требований для всех учащихся.

При оценивании результатов обучения стоит учитывать, что оно может производиться на разных уровнях (рисунок 1):

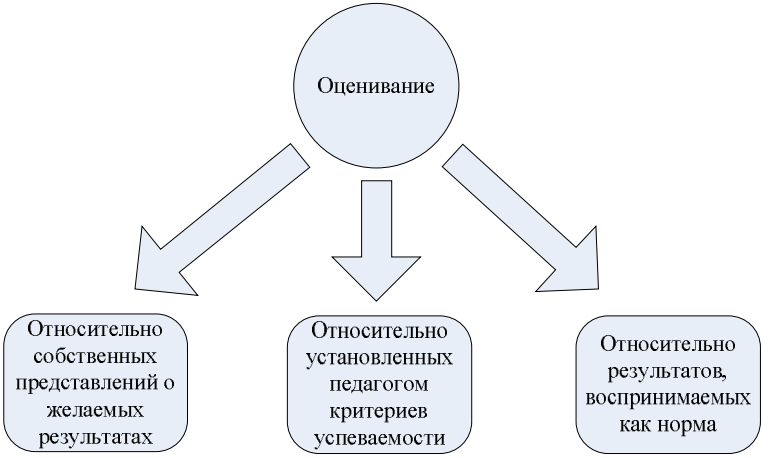


Рисунок 1 – Уровни оценивания результатов обучения

Первым методом контроля результатов обучения становится устный контроль. Устное повторение в классе, даже на первом вводном занятии, помогает определить общее понимание темы обучающимися, но может быть усилено с помощью электронного портфолио. Новый материал, объясненный преподавателем, обычно записывается обучающимися в тетрадь. Тогда у них появляется возможность позднее повторить и переосмыслить пройденное. Электронное портфолио позволяет хранить этот материал на сайте, в приложении или в облаке, а запись может принимать различные формы, включая легко запоминающиеся ментальные схемы, мнемонические фразы, зарисовки и т.п.

Письменный контроль может осуществляться как на занятии, так и в виде сданных работ. Его основной недостаток состоит в том, что при традиционных методах оценивания все работы сдаются учителю на бумаге, и увидеть их может только он. Технология электронного портфолио позволяет оцифровывать подобные работы, подключать комментирование другими обучающимися. Письменные задания возвращаются ученикам после проверки, а затем снова поступают к учителю для доработки и исправлений. После этого работы нередко остаются у педагога, что мешает обучающимся оценить эффективность своих доработок и определить направление совершенствования навыков в будущем. Этого можно избежать, если оцифровать задания и хранить их в облачном хранилище портфолио. Обучаемые смогут в любое время обратиться к своей работе и убедиться, что все ошибки исправлены, а их причины усвоены. Кроме того, письменные задания, созданные и сохраненные с помощью технологии электронного портфолио, интерактивны и могут использовать мультимедийные технологии (например, аудио, видео, ссылки на веб-ресурсы).

Наиболее важной особенностью электронного портфолио для осуществления практического контроля является то, что его можно вести вне аудиторных занятий, а данные передавать в электронном виде. Любые результаты экспериментов или практической работы могут быть сохранены в облачном хранилище. Результаты, сохраненные таким образом, могут быть отправлены другим учителям для оценки, использования в конспектах или пересмотра.

Таким образом, можно сделать вывод, что технология электронного портфолио дополняет и улучшает традиционную систему оценивания. Главное преимущество электронного портфолио как инструмента оценивания – это широкие возможности для хранения заданий, комментариев, выводов и обсуждений в одном месте и

использования их в более позднее время без необходимости их потери. Учителю не нужно полагаться только на обезличенную оценку представленных учениками работ для всесторонней оценки их работы за определенный период времени, он может иметь более полное представление об этих заданиях, оценках, исправлениях и комментариях для анализа и оценки работы своих учеников. Технология электронного портфолио также позволяет исправлять ошибки в тех случаях, когда это невозможно при традиционной системе оценивания. Например, ошибки во время экзамена могут быть исправлены учителем и повлиять на оценку, но не после экзамена, что может привести к отсеву. Представление, обсуждение в классе и хранение задания, результатов и комментариев в личном пространстве пользователя позволяет собрать и исправить все ошибки и упущения в одном месте, где они могут быть рассмотрены учителем и другими учениками.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Сапрыкина, А.О. Анализ влияния электронного портфолио обучающихся на процесс трудоустройства // Материалы XXVII Всероссийской научно-технической конференции студентов, молодых ученых и специалистов «Новые информационные технологии в научных исследованиях». – Рязань: РГРТУ, 2022. – Том 1. – С. 74-76.

2. Сапрыкина, А.О. Причины снижения эффективности использования электронного портфолио в образовательном процессе // Материалы XXVI Всероссийской научно-технической конференции студентов, молодых ученых и специалистов «Новые информационные технологии в научных исследованиях». – Рязань: РГРТУ, 2021. – С. 46-47.

УДК 004. 032.26

М.Н. САРАЕВ, С.И. БАБАЕВ
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ИССЛЕДОВАНИЕ АЛГОРИТМА НЕЙРОСЕТЕВОЙ МОДЕЛИ

В статье рассматриваются вопросы по анализу данных с помощью свёрточных нейронных сетей и процесса моделирования алгоритмов нейросети на основе модели биологических нейронов.

Актуальность данной темы заключается в том, что существует проблема накопления большого объёма информации и массива данных, который нужно обрабатывать, приводить в упорядоченное состояние, что возможно только благодаря современным технологиям в области нейросетей и искусственного интеллекта. Основная задача нейронных сетей – это распознавание данных, прогнозирование и управление сложными системами, те задачи, которые ранее мог выполнять только человек. Уже сейчас с некоторыми задачами «машины» лучше справляются, чем человек, и со временем данная тенденция будет только набирать силу, количество будет переходить в качество управления.

Нейросеть и биологический нейрон

Эпоха появления искусственных нейронных сетей начинается с 1940-х годов с появлением фундаментальных понятий кибернетики и представлении сложных биологических процессов формальными математическими методами. На протяжении более 80-ти лет лидирует идея нейроцентричности, при которой главенство базового элемента «когнитивной архитектуры» занимают нейроны. В настоящий момент времени нейросети это сильно упрощенные модели биологической нервной ткани, которые представляют собой сети, состоящие из нейронов, например в виде графов: где вершины – это непосредственно сами нейроны, а ребра – синаптические связи. [1]

Основная идея нейросетей, которая была «подсмотрена» в живой природе, это принципы функционирования нейрона и совместное взаимодействие групп нейронов. Например, в коре головного мозга млекопитающих нейроны образуют группы в виде мини колонок кортекса по 80-120 нейронов, при этом каждый нейрон может иметь до 10 тысяч синапсов. Это элементарные и устойчивые структуры функционирования мозговой активности.

Тело нейрона по своей сути имеет наружную поверхность, состоящую из мембраны, которая способна как выпускать из себя (из клетки) наружу ионы, так и запускать в себя ионы натрия, калия, хлора, магния, кальция. Это происходит за счёт механизма активации ионных каналов и диффузионного потока ионов, а также ионных насосов, которые представлены на рисунке 1. За короткий промежуток времени такое быстрое движение потоков ионов формирует кратковременное изменение мембранного потенциала на небольшом участке тела возбудимой клетки нейрона, таким образом происходит передача возбуждения в виде потенциала действия. По времени на формирование потенциала действия уходит около 5 мс, поэтому биологический нейрон способен генерировать не более 200

потенциалов действия в секунду, в чем уступает искусственным нейронам, которые такого ограничения не имеют [2].

Семантическая и математическая модель нейрона

Опишем семантическую модель работы биологического нейрона в виде следующей архитектуры:

- принятие входящих потенциалов действия от других нейронов на мембране конкретного нейрона;
- происходит процесс вычисления входящих сигналов на мембране нейрона;
- при достаточной силе сигнала нейрон передаёт возбуждение дальше по своему аксону тем нейронам, с которыми образовались устойчивые связи.

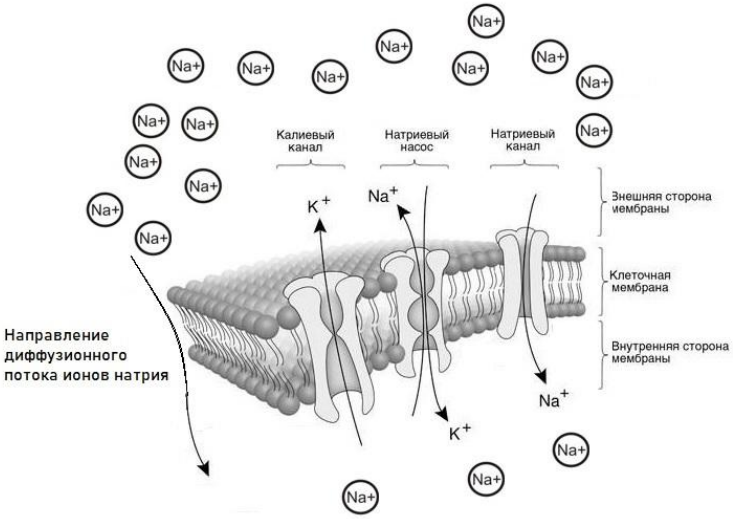


Рисунок 1 – Фрагмент мембраны тела биологического нейрона

Тогда формальная математическая модель будет включать в себя суммирование взвешенных входящих сигналов $\mathbf{w_{kj}x_j}$ на «мембрану» искусственного нейрона **«к»** и преодоление «барьера» активационной функции **φ** для передачи выходного сигнала $\mathbf{y_k}$, где **n** – количество входящих синаптических сигналов. [3] Передача сигнала не состоится, если не будет преодолена величина порога активационной функции:

$$y_k = \varphi\left(\sum_{j=0}^{n} w_{kj} x_j\right).$$

Персептрон и свёрточная нейронная сеть

Рассмотрим реализацию свёрточной нейронной сети, представленную на рисунке 2. Такая сеть имеет вид персептрона, который обладает входным слоем нейронов (**input),** принимающих вектор признаков или атрибутов, в нашем случае в виде матрицы пикселей некоторого изображения. Далее находятся слои свертки (**convolutional layer)**, которые выполняют всю «умную» работу по обработке и распознаванию данных, например, в виде изображения объектов на исходной картинке. Заключительный – выходной слой (**output**) выводит некоторый искомый результат, например, определяет, к какому классу относится изображение [4].

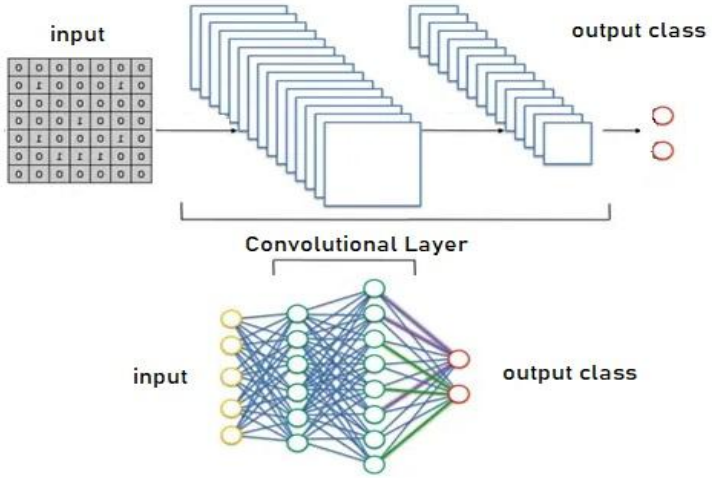


Рисунок 2 – Свёрточная нейронная сеть

Основная идея в сверточной нейронной сети заключается в использовании матрицы весов, которую называют ядром свертки. Она в свою очередь интерпретирует данные в виде графического кодирования какого-либо признака. Последующий слой формирует в результате операции свертки карту признаков, например, в форме линии или дуги, расположенные под различными углами. Следующий слой нейронной сети может формировать свою карту признаков с более сложными формами данных, например, более сложные графические признаки.

Для того чтобы сверточные нейронные сети могли успешно распознавать множество различных объектов на изображении, необходимо достаточно большое количество свёрточных слоёв. Так что бы нейросеть могла распознавать интерьер квартиры, обстановки и мебели, а также ландшафта окружения дома, необходимо более 13-ти свёрточных слоев [5].

Проблемы, которые возникают во время разработки и исследования нейронных сетей, лежат как в плоскости поиска оптимальной системы и структуры нейросети, так и в плоскости процесса последующего её обучения и переобучения. Уже сформированную, обученную и зарекомендовавшую себя нейросеть вполне возможно адаптировать под новые задачи, расширяя тем самым её функционал и применимость. Горизонт данной задачи лежит как в области архитектуры алгоритмов нейросетей, так и в аппаратном обеспечении, новых типах компонентной базы, поэтому проблемы нейросетей в высокой степени относятся к междисциплинарным научным задачам.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Artificial neuron. – Текст: электронный. – URL: https://en.wikipedia.org/wiki/Artificial_neuron (дата обращения: 08.04.2023)

2. Потенциал действия. – Текст: электронный. – URL: https://en.wikipedia.org/wiki/Action_potential (дата обращения: 08.04.2023)

3. Корячко В.П. Интеллектуальные системы и нечеткая логика: учебник / В.П. Корячко, М.А. Бакулева, В.И. Орешков – Москва: Изд-во Курс, 2017. – 352 с.

4. Хайкин С. Нейронные сети: учебник: полный курс, 2-е издание – М. Изд-во «Вильямс», 2006. – 1104 с.

5. David B. Understanding the role of individual units in a deep neural network. – Текст: электронный. – URL: https://doi.org/10.1073/pnas.1907375117 (дата обращения: 08.04.2023).

УДК 004

Е.Ю. СКОЗ, П.С. СКОЗ
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

НОВЫЕ ВИДЫ КИБЕРАТАК

Рассматриваются способы имитации атак извне информационной системы с применением аппаратуры, для усиления защиты данных.

В настоящее время организация режима информационной безопасности становится критически важным стратегическим фактором развития.

Сейчас все чаще в информационных источниках встречается понятие системного подхода при построении средств защиты информации (СЗИ). Понятие системности заключается не просто в создании соответствующих механизмов защиты, а представляет собой регулярный процесс, осуществляемый на всех этапах жизненного цикла информационных систем. Многообразие вариантов построения информационных систем порождает необходимость создания различных систем аппаратной защиты, учитывающих индивидуальные особенности каждой из них.

Среди криптографических программных средств с платной лицензией здесь можно выделить продукты eToken и ruToken среди свободно распространяемых программ – PGP, TrueCrypt. Однако данные программы обладают рядом недостатков – они либо платные, либо являются продуктами иностранного производства, где не гарантировано отсутствие закладок.

Появление все более новых мощных компьютеров, технологий сетевых и нейронных вычислений сделало более достижимой дискредитацию криптографических систем еще до недавнего времени считавшихся практически не раскрываемыми.

В большинстве случаев, на некоторых этапах своего жизненного цикла информация оказывается вне зоны непосредственного контроля над ней. Физически предотвратить внесение несанкционированных изменений в данные в подавляющем большинстве реальных систем их обработки, передачи и хранения невозможно. Поэтому весьма важно вовремя обнаружить сам факт таких изменений.

Таким образом, под защитой данных от несанкционированных изменений понимают не столько исключение возможности таких изменений, а набор методов, позволяющих надежно зафиксировать их факты, если они имели место.

Достаточно полный профиль системы можно получить, используя содержимое рабочей памяти одного процесса, ядра или всей операционной системы (дамп памяти). Существует случаи, когда снимки памяти очень пригодились бы злоумышленнику: метод холодной перезагрузки и атака через DMA.

1. Атака при помощи метода холодной перезагрузки (Cold boot attack).

В начале 2008 года был описан новый вид атак на работающие системы (live systems), целью которых является получение

информации из оперативной памяти. Этот вид атак основан на получении информации, которая остается в оперативной памяти. Считается, что данные в оперативной памяти после выключения компьютера немедленно теряются.

Однако исследования показали, что это не совсем так. На самом деле, требуется некоторое время для очистки данных из памяти. При отключении питания компьютера большая часть информации на секунду или две остается в целости и сохранности.

Это время можно увеличить путем охлаждения памяти, используя следующий прием: опрыскивание платы памяти, баллончиком со сжатым воздухом, перевернутым верх дном. Охлаждение платы позволяет увеличить время, в течение которого данные могут оставаться в памяти – до нескольких минут, что позволило исследователям разработать серию инструментов, которые могут извлекать информацию из памяти на машинах, где произошло быстрое выключение и повторное включение питания.

При использовании этого метода появляется существенная вероятность получить небольшие ошибки внутри снимка памяти, из-за чего его избегают использовать, например, эксперты-криминалисты. Однако злоумышленнику не требуется образ памяти, соответствующий высоким уровням стандарта. При сохранении ключей шифрования и баз данных SAM, даже 2% повреждений вполне допустимо, что позволяет использовать этот метод для внутреннего тестирования на проникновение.

Исследователями был разработан комплект утилит для реализации атаки методом холодной перезагрузки.

Защита от атаки методом холодной перезагрузки может быть реализована размещением в корпусе сервера блока бесперебойного пита-ния и установкой датчиков вскрытия корпуса.

Также возможно комбинирование датчиков вскрытия корпуса и устройства по уничтожению информации на магнитных носителях. В зависимости от уровня важности информации и наличия резервных копий.

2. Атака через Firewire.

IEEE 1394 интерфейс, продвигаемый компанией Apple как FireWire, - высокоскоростной коммуникационный интерфейс, который первоначально задумывался в качестве замены SCSI. Главная отличительная черта FireWire – высокоскоростная передача информации. Именно поэтому данный протокол преимущественно используется для передачи больших объемов видео- и аудио-данных. Непосредственное обращение к памяти через DMA (Direct Memory

Access), полностью минуя CPU – одна из особенностей FireWire, которая позволяет передавать информацию на высокой скорости. Это свойство хорошо и для приложения по видеомонтажу при загрузке фильмов с видеокамеры, и для злоумышленника.

Прямой доступ памяти в FireWire позволяет злоумышленнику получить физический доступ к целевой машине для обхода парольной защиты в операционной системе путем перезаписи области памяти, содержащей функции по контролю за доступом. DMA также позволит злоумышленнику загрузить первые 4 ГБ оперативной памяти. Подобный вид доступа представляет серьезную уязвимость в спецификации интерфейса IEEE 1394, хотя у FireWire есть доступ только к первым 4 ГБ оперативной памяти и этот интерфейс может быть защищен антивирусом для предотвращения доступа через DMA. Большинство операционных систем пытаются ограничить доступ к DMA только для известных устройств, но подобную защиту относительно легко обойти и подделать эти устройства.

Несмотря на то, что в основном подобные атаки направлены на FireWire, их можно реализовать на любом устройстве, которое использует подобную шину: ExpressCard (EC), PC Card и интерфейс Thunderbolt.

Была создана утилита под названием AESKeyFind для анализа снятых образов оперативной памяти, которая искала развертки AES-ключей и извлекала эти ключи шифрования из памяти. Даже если образ испорчен или содержит ошибки, утилита сможет достать ключи.

Была также создана еще одна утилита – Volatility, которая представляет собой фреймворк с открытым исходным кодом для выполнения криминалистического анализа. У Volatility есть несколько модулей, позволяющих злоумышленнику извлекать полезную информацию о за-пущенных процессах и настройках целевой машины. Если образ не содержит ошибок, злоумышленник даже может извлечь Windows-хеши.

3. DMA атака.

DMA атака – это атака на чипсет. У современных чипсетов имеется контролер DMA, который используется для разгрузки центрального процессора от задач пересылки данных из памяти и в память, как между DRAM, так и между DRAM и памятью PCIe-устройств. Причем инициировать такую передачу может как ОС, так и PCIe-устройство, у которого тоже может быть свой DMA-контролер. Причем, в случае инициации из ОС, операцию доступа к данным осуществляет чипсет, т.е. любые настройки процессора на него если и влияют, то достаточно слабо. Т.е. атакующий с правами

администратора инициирует DMA с контролируемой им областью памяти с одной стороны и SMRAM с другой, и получает полный доступ к ней. То же самое может сделать и прошивка контролируемого атакующим PCIe-устройства. Для защиты от DMA-атаки у DMA-контролеров Intel имеется регистр TSEGMB в котором дублируется информация о местонахождении TSEG, и бит Lock, который запрещает это самое содержимое менять. Если авторы прошивки забыли установить – можно атаковать.

В оперативной памяти хранится достаточное количество информации, используя которую злоумышленник может получить доступ к конфиденциальным данным или даже доступ ко внутренней сети. Подобная проблема никогда не принималась во внимание, поскольку не было практического способа получения образа памяти со стороны внешнего источника. Однако атака методом холодной перезагрузки и атака через DMA делают возможным не только получение доступа к па-мяти, но и получение доступа к секретным ключам.

В качестве дополнительных мер борьбы с несанкционированным доступом можно использовать устройство уничтожения информации встраиваемое прямо в компьютерный корпус, которое может срабатывать при несанкционированном вскрытии корпуса, либо при активации дистанционно. Схема устройства дистанционной активации показана на рисунке 1.

Рисунок 1 – Схема устройства дистанционной активации

Уничтожители информации в серверных комнатах встраиваются в серверные шкафы, либо для них изготавливается напольный корпус. Срабатывание системы уничтожения данных может происходить при несанкционированном открытии дверей в серверную комнату или серверный шкаф, а также при несанкционированной выемке дисков.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Чиликов А., Алексеев Е. Поиск криптографических ключей в RAM. – Текст: электронный. – URL: http://www.ruscrypto.ru/ conference/program/reversing/ (дата обращения: 15.04.2023).

2. Олег Зензин. Режимы шифрования. – Текст: электронный. – URL: http://www.citforum.ru/security/cryptography/rejim_shifrov (дата обращения: 15.04.2023).

3. PGP. – Текст: электронный. – URL: http://ru.wikipedia.org/wiki/PGP (дата обращения: 15.04.2023).

УДК 004.032.26

С.С. ТУРНАЕВ
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

ИСКУССТВЕННАЯ НЕЙРОННАЯ СЕТЬ В МАШИННОМ ОБУЧЕНИИ

Рассматриваются основные определения, связанные с искусственной нейронной сетью, а также ее структура и основные отличия от обычной компьютерной программы, которые позволяют ей решать более сложные задачи.

Нейронная сеть в машинном обучении, строится по принципу работы человеческого мозга, который состоит из миллионов нейронов. Мозг посылает и обрабатывает сигналы в виде электрических и химических сигналов. Передавать сигналы между собой нейронам позволяет специальная структурная связь, которая называется синапсом. Из большого такого количества нейронов формируются нейронные сети.

Искусственные нейронные сети представляют собой вычислительные алгоритмы, которые используются для имитации поведения биологических систем, состоящих из нейронов. Модель, состоящая из таких алгоритмов должна быть способна к машинному обучению, а также распознаванию образов, полученных на основе входных данных. Математическая модель представляет собой ориентированный граф, каждый узел которого имеет свой весовой коэффициент. Искусственная нейронная сеть обычно состоит из трех слоев, которые в свою очередь состоят из множества взаимосвязанных узлов (рисунок 1).

Входной слой – это пассивный слой, он получает данные для каждого наблюдения и, не обрабатывая их, дублирует значения на множество своих выходов.

Скрытый слой – слой, который производит заданные преобразования входных переменных. Обработка выполняется через систему взвешенных соединений. Таких слоев может быть несколько.

Выходной слой – он возвращает выходное значение, соответствующее предсказанию переменной ответа. Правильность полученного результаты во многом зависит от правильного выбора весов.

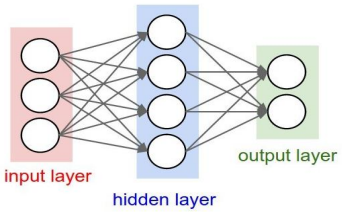


Рисунок 1 – Структура искусственной нейронной сети

Нейронная сеть, рассмотренная выше называется многослойной, но существует и более простой вид сети, именуемой однослойной. Такая структура представляет собой сеть, в которой сигналы со входного слоя сразу попадают на выходной слой. Входной слой принимает и распределяет сигналы, а вычисления происходят в выходном слое. Такие структуры проще в реализации и могут применяться для простых вычислений.

Кроме количества слоев, нейронные сети можно классифицировать по распределению информации по синапсам:

1. Однонаправленные – сигналы перемещаются строго по направлению от входного сигнала к выходному. Движение сигнала в обратном направлении невозможно.

2. С обратными связями – сигналы двигаются и в прямом, и в обратном направлении. Результат с выхода, способен снова оказаться на входе.

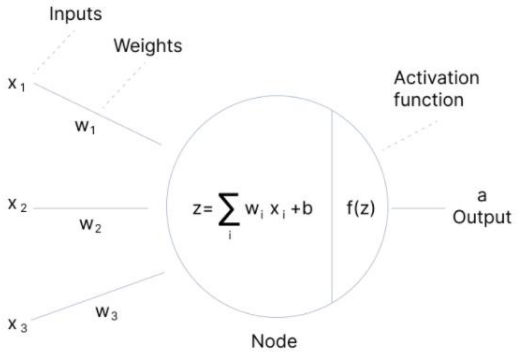


Рисунок 2 – Упрощенная схема обработки входных сигналов

Человеческий мозг, анализируя входные данные, обрабатывает и решает, должен ли нейрон быть активирован, в искусственной нейронной сети эту роль выполняет функция активации. Основная ее роль заключается в преобразовании суммированного взвешенного ввода от узла к выходу, либо следующему узлу. Различают три типа функций активации:

1. Ступенчатая функция активации – входное значение сравнивается с определенным пороговым значением, и принимается решение о том, будет нейрон активирован или нет.

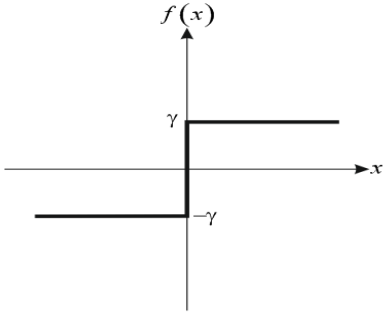


Рисунок 3 – Ступенчатая функция активации

2. Линейная функция активации – входное значение пропорционально выходному, однако у этой функции есть два больших недостатка: невозможность реализовать обратную связь, так как производная этой функции константа, а также независимо от количества скрытых слоев, выходной слой будет являться линейной функцией входного слоя.

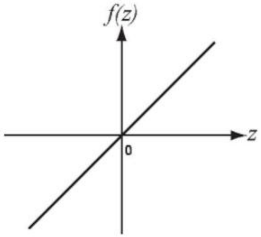


Рисунок 4 – Линейная функция активации

3. Нелинейные функции активации – эти функции позволяют реализовать обратную связь для того чтобы можно было корректировать весовые коэффициенты для достижения лучшего результата. Также, в отличие от линейной функции, появляется

возможность использовать множество слоев, которые будут накладываться друг на друга. На рисунке ниже представлены нелинейные функции активации.

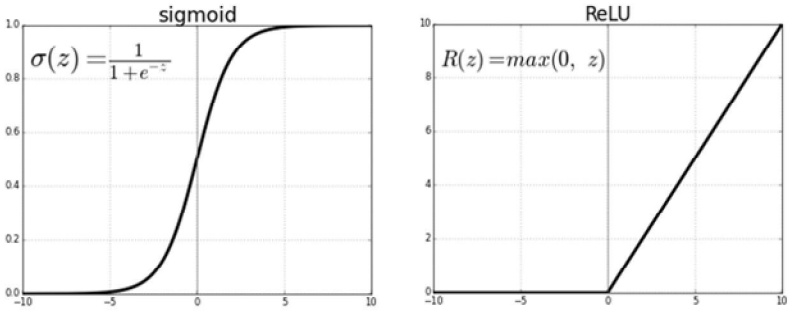


Рисунок 5 – Нелинейные функции активации
(сигмоид и усеченная функция)

Искусственные нейронные сети применяются для решения сложных задач, с которыми простые компьютерные программы справится не могут. Такими задачами, могут быть распознавание образов, предсказание следующего шага (в основном используются на фондовом рынке), классификация входной информации по определенным параметрам – так работают кредитные роботы, которые позволяют быстро принять решением об одобрении или отказе кредита.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Хайкин, Саймон. Нейронные сети: полный курс, 2-е издание: Пер. с англ. – М.: Издательский дом «Вильямс», 2006.

2. Гафаров Ф.М., Галимянов А.Ф. Искусственные нейронные сети и приложения. – Казань, 2018.

3. Гудфеллоу Я., Бенджио И., Курвилль А.. Глубокое обучение, 2-е изд.: пер. с англ. А.А. Силкина. – М: ДМК Пресс, 2017.

4. Васильев А.Н., Тархов Д.А.. Нейросетевое моделирование. Принципы. Алгоритмы. Приложения. – СПб.: Изд-во Политехн. Ун-та, 2009.

5. Тархов Д.А. Нейронные сети. Модели и алгоритмы. – М., Радиотехника, 2005.

УДК 004.415.2.031.43

З.А. ХАХАЛИН
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

РЕАЛИЗАЦИЯ ДЕТЕКТОРА ОБЪЕКТОВ РЕАЛЬНОГО ВРЕМЕНИ С ИСПОЛЬЗОВАНИЕМ НЕЙРОСЕТЕВОГО ПОДХОДА YOLO И СРЕДСТВ OPENCV

В статье рассматривается реализация детектора автотранспортных средств реального времени, основанного на нейросетевом подходе YOLO и инструментах библиотеки компьютерного зрения OpenCV.

Алгоритмы детектирования объектов интереса на видеопоследовательности лежат в основе многих систем захвата и сопровождения (трекинга). В их задачи входит обнаружение и классификация объектов на изображении, определение их координат и координат их ограничивающих рамок. Число потенциальных объектов интереса на каждом кадре неизвестно, при этом классы объектов известны заранее.

Наиболее эффективные алгоритмы детектирования, разработанные в последнее время, используют нейронные сети и реализуют одноступенчатый принцип, когда используется один проход входного изображения через нейронную сеть для прогнозирования наличия и местоположения объектов. Такой подход впервые был реализован в технологии YOLO (англ. – “You only look once”) в 2015 году и показал кратный прирост производительности в сравнении с другими алгоритмами (до 160 кадров в секунду) [1].

Принцип работы детектора YOLO

Исходное изображение разбивается на сетку из S×S ячеек (рисунок 1). Если объект попадает в ячейку, она объявляется ответственной за определение параметров его местоположения. В ячейке генерируется несколько ограничивающих рамок (обычно 3) для одного объекта. Каждая рамка описывается вектором значений – координатами центра, шириной, высотой и степенью уверенности наличия объекта ней. Также определяется вероятность наличия объекта этого класса в ячейке. Таким образом, последний слой сети, принимающий конечное решение об ограничивающих рамках и классификации объектов работает с тензором размерности S×S×(5B+C), где B – количество предсказываемых ограничивающих рамок для ячейки, C – количество классов объектов, определённых изначально.

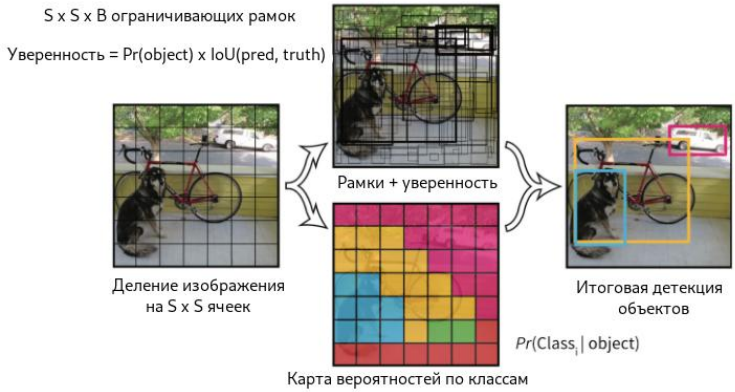


Рисунок 1 – Принцип работы детектора YOLO

Нейронная сеть работает следующим образом (рисунок 2). Входное изображение проходит через 24 слоя свертки и подвыборки, формируя некоторый набор признаков. Получившийся набор передается на 2 последовательных полносвязных слоя, где происходит предсказание выходных вероятностей и координат ограничивающих рамок. Чередование слоев редукции 1×1 со слоями 3×3 позволяет дополнительно уменьшить пространство признаков на каждом слое.

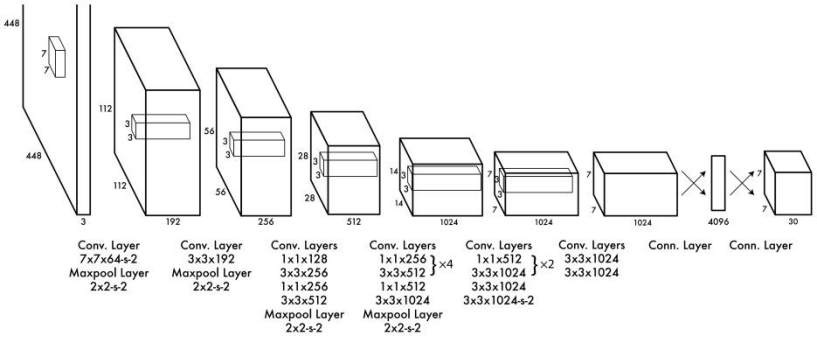


Рисунок 2 – Архитектура нейронной сети детектора YOLO

Реализация детектора автотранспорта с использованием YOLO и OpenCV

Для реализации чтения и записи кадров использован функционал библиотеки с открытым исходным кодом OpenCV, а также библиотеки стандарта сжатия видео H.264 (таблица 1).

Таблица 1. Используемые технологии

Название	Описание	Версия
YOLO	Технология детектирования объектов	v3, v3-tiny
OpenCV	Библиотека компьютерного зрения	4.4.0.0
EmguCV	.NET-оболочка над OpenCV	4.4.0.4061
.NET SDK	Технология для разработки приложений на языке C#	7.0.202
NVidia cuDNN	Библиотека примитивов для глубоких нейронных сетей с ускорением на графическом процессоре	8.0.1
H.264	Лицензируемый стандарт сжатия видео, предназначенный для достижения высокой степени сжатия видеопотока при сохранении высокого качества	1.8.0

Нейронная сеть обучается на изображениях размера кратных 32x32, поэтому каждый новый кадр должен быть приведен к соответствующему размеру. Для конфигурации YOLO предусмотрены две коэффициента – порог подавления не-максимумов и порог коэффициента доверия. Первый позволяет отфильтровать ложные детекции объекта интереса. Второй отражает вероятность наличия объекта интереса в ограничивающей рамке, и точность определения этой рамки.

В таблице 2 приведены результаты замера средней частоты кадров для обычной и облегченной версий YOLO (yolov3 и yolov3-tiny, соответственно) при изменении разрешения входного изображения.

Таблица 2. Результаты замера средней частоты кадров

Разрешение	yolov3-tiny (CPU)	yolov3-tiny (GPU)	yolov3 (CPU)	yolov3 (GPU)
320 x 320	50	**200**	5	33
640 x 640	14	45	1,4	14
960 x 960	7	35	0,6	7
1280 x 1280	4	23	**0,3**	4

Усредненная по категориям величина средней точности детекции (mAP) для yolov3 составляет 57,9%, для yolov3-tiny – 33,1%. По этому показателю YOLO уступает другим методам, но в плане производительности опережает всех [3].

Для визуализации видеообработки и возможности быстрого конфигурирования параметров YOLO разработан программный стенд (рисунок 3).

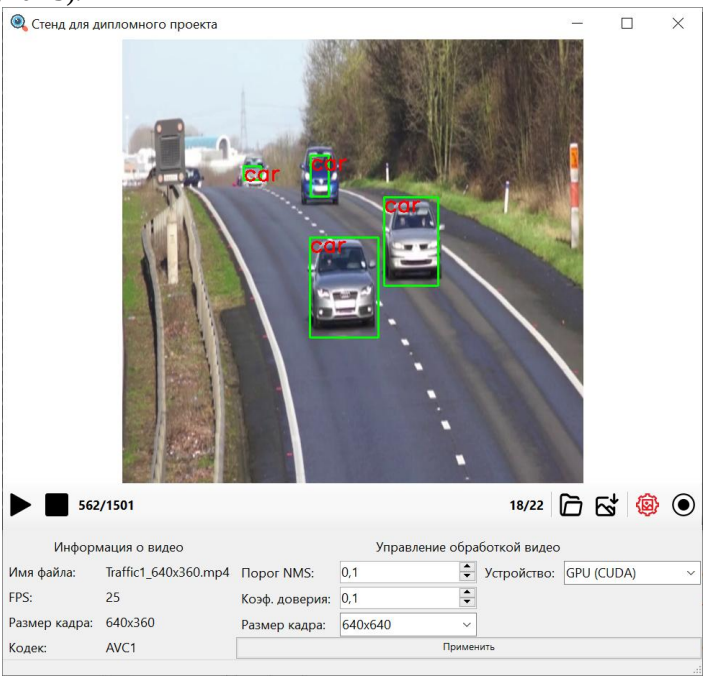


Рисунок 3 – Программный стенд для детектирования автотранспорта

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Zou Z. Object Detection in 20 Years: A Survey / Z. Zou, K. Chen, Z. Shi, Y. Guo, J. Ye // Proceedings of the IEEE. – 2023. – Vol. 111, no. 3. – Pp. 257-276.

2. Redmon J. You only look once: unified, real-time object detection / Redmon J., Divvala S., Girshick R., Farhadi A. // IEEE conference on computer vision and pattern recognition. – 2016. – Pp. 779-788.

3. Performance on the COCO Dataset. – Текст: электронный. – URL: https://pjreddie.com/darknet/yolo/ (дата обращения: 11.04.2023).

УДК 004.9

А.А. ХРАМОВА, А.Н. САПРЫКИН
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

РАЗРАБОТКА ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА ВЕБ-ОРИЕНТИРОВАННОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ

Рассматривается полный цикл разработки и принципы проектирования пользовательского интерфейса веб-ориентированной информационной системы.

Пользовательский интерфейс – это граница между двумя функциональными объектами, пользователем и системой. Под понятием пользовательского интерфейса информационной системы (ИС) обычно понимают внешний вид программы, однако не стоит забывать, что он включает в себя такие составляющие, как набор задач пользователя, навигацию между блоками системы, визуальный дизайн ИС и т.д.

Рассмотрим полный цикл разработки пользовательского интерфейса веб-ориентированной ИС:

- анализ предметной области;
- написание пользовательских сценариев;
- проработка структуры интерфейса;
- разработка макета;
- выбор стиля интерфейса;
- разработка концепции дизайна;
- оформление всех экранов;
- добавление анимации к элементам;
- разработка интерфейса программистами.

Рассмотрим каждый из этапов разработки интерфейса в отдельности.

Анализ предметной области

Данный этап представляет собой сбор информации об ИС, ее пользователях и аналогах (конкурентах). Если ИС уже имеет пользовательский интерфейс, то проводится анализ статистики использования текущего интерфейса, а также анализ устройств предполагаемой целевой аудитории.

Определяются ограничения, которые следует учесть при проектировании: интерактивность, адаптация под различные устройства (смартфоны, планшеты, ноутбуки и т.д.).

Написание пользовательских сценариев

На данном этапе создается список задач, которые может выполнять клиент при использовании интерфейса ИС.

Пользовательский сценарий – это план действий клиента, представленный в виде образа идеального пользователя, пытающегося достичь своей цели в ходе взаимодействия с ИС. Чтобы представить пользователей, сценарии должны исходить из четкого понимания того, кто будет пользоваться ИС, для этого на предыдущем этапе определяется возможная целевая аудитория.

При работе с хорошо продуманными пользовательскими сценариями команда разработчиков может раньше выявить проблемные области и затем исправить их. С помощью них можно определить этапы, на которых у пользователей могут возникнуть трудности, а также сократить путь решения определенных задач.

Проработка структуры интерфейса

Структура интерфейса создается по полученным на предыдущем этапе пользовательским сценариям. Определяется количество страниц интерфейса, их положение в общей структуре, а также краткое содержание каждого экрана. Она демонстрирует логику, дорожную карту взаимодействий с интерфейсом и его состояние на каждом шаге.

Разработка макета

Макет – это статическое визуальное представление концепции пользовательского интерфейса. Макеты используются для обсуждения функциональности ИС с заказчиком, для постановки задачи разработчикам и дизайнерам.

Преимущества макетов:

- оперативность внесения правок без лишней перерисовки;
- минимизация или полное исключение правок на последующих этапах разработки;
- объективная оценка сроков и стоимости разработки пользовательского интерфейса;
- визуализация промежуточных этапов и четкое понимание дальнейших шагов.

Макеты делятся на две группы: wireframes (макеты низкой точности или вайрфреймы) и mockups **(**макеты высокой точности или мокапы).

Wireframes используют для схематичного изображения элементов, чтобы обозначить структуру интерфейса (разметка «Header», «Footer» и т.д.). Они помогают проектировщику понять объем интерфейса, как лучше расположить элементы, чтобы избежать их переизбытка. Wireframes не заостряют внимания на таких аспектах

дизайна как цвет и типографика. Зачастую они выполняются в черно-белом цвете.

Mockups в свою очередь представляют приближенный к реальности вариант внешнего вида пользовательского интерфейса. Мокапы бывают разной степени достоверности. Самые достоверные полностью соответствуют финальному виду интерфейса (до пикселя) и, как правило, разрабатываются профессиональными дизайнерами.

Создание макетов любого вида облегчает взаимопонимание между заказчиком и разработчиками, а также делает разработку ИС экономичной.

Выбор стиля интерфейса

Параллельно с этапами создания макетов интерфейса идет определение его стилистики. Рассмотрим наиболее популярные стили для создания интерфейса.

Скевоморфизм. Его концепция заключается в повторении реалистичных форм и видов объектов из настоящей жизни (свет, тени, блики, текстуры и т.д.) в виртуальном пространстве.

Metro. Основными принципами данного стиля являются: высокая читаемость, легкое восприятие информации и отсутствие отвлекающих факторов. Под стиль Metro был разработан уникальный шрифт – Segoe.

Flat Design. Данный стиль отличается от скевоморфизма тем, что элементы «плоские». У них отсутствуют блики, тени и текстуры.

Material. Этот стиль является обобщением стилей скевоморфизм и Flat Design. Он использует «плоские» элементы, но при этом некоторые из них имеют теневой эффект.

Минимализм. Его концепция заключается в создании простых для восприятия, лаконичных и эффективных интерфейсов. Характерными чертами минимализма являются:

- свободное пространство;
- минимальное количество используемых шрифтов и цветов;
- минимальное количество визуального шума (тени, графика и т.д.).

Неоморфизм. Данный стиль напоминает скевоморфизм тем, что элементы имеют выпуклую форму, повторяя тем самым объекты в реальной жизни, но в неоморфизме объем элементов создается исключительно с помощью теней. Визуально данный способ создания выпуклости элементов делает интерфейс гораздо «чище».

Разработка концепции дизайна

Концепция дизайна необходима для демонстрации оформления интерфейса и его будущего вида. Предыдущий этап выбора

стилистики интерфейса определяет только направление, а концепция дизайна позволяет совместить выбранный стиль с имеющимся содержанием интерфейса.

Для заказчика концепция дизайна интерфейса может быть представлена любым объемом, но зачастую его стараются минимизировать до 1-3 экранов интерфейса для экономии времени.

Оформление всех экранов

После утверждения концепции дизайна с заказчиком оформляются остальные экраны интерфейса.

На предыдущем шаге определялось, как может выглядеть весь интерфейс. На шаге оформления всех экранов происходит финализация внешнего вида интерфейса: правильно ли подобран кегль или расстояние между строками, сочетается ли оформление форм, таких как кнопки, поля ввода с другими элементами экрана и т.д.

Планом для оформления всех экранов являются структура и макет интерфейса. Все оформленные экраны собираются в интерактивный прототип, который создает максимально приближенный вид использования интерфейса без использования услуг разработчиков. Создание прототипа экономит денежные затраты на разработку непроверенного функционала. И даёт возможность посмотреть на «рабочее приложение» до того, как макеты интерфейса будут переданы разработчикам.

Добавление анимации к элементам

Анимация позволяет сделать интерфейс интерактивным, с отсылкой к реальному миру. С помощью нее можно подсказывать пользователю следующее действие, объяснять связь между предыдущим и текущим состоянием и давать ощущение контроля над интерфейсом.

Для правильного использования анимации следует соблюдать ключевые принципы и правила. Рассмотрим некоторые из них.

Длительность и скорость анимации. При создании эффекта перемещения элементов необходимо найти «золотую середину» в длительности. Анимация должна быть достаточно медленной, чтобы пользователь заметил движение, но в то же время достаточно быстрой, чтобы не заставлять его ждать.

Согласно руководству по материальному дизайну, для мобильных устройств рекомендуется устанавливать длительность анимации 200-300 мс. На планшетах длительность должна быть на 30% больше – 400-450 мс. Разница в длительности анимации зависит от размера устройства, т.к. на большом устройстве объект будет проходить больший путь до своего места назначения. На

персональных компьютерах, анимация должна быть примерно в 2 раза короче, чем на мобильных устройствах – от 150 до 200 мс, чтобы не создавать эффекта зависания устройства.

Сглаживание. Эффект сглаживания придает объекту естественность, характер и динамику. Обычно сглаживание описывается с помощью графиков зависимости дистанции элемента (ось y) и времени на ее прохождение (ось x).

Линейное движение. Если на объект не действует физическая сила, то он движется с постоянной скоростью – линейно. У пользователя может сложиться впечатление, что о нереальности объекта (может показаться ненастоящим, искусственным). Линейное движение зачастую используется в тех случаях, когда объект не меняет позицию, а меняется только его цвет или прозрачность.

Плавное ускорение. Данная анимация применяется для нарастания скорости движения объекта. Для создания плавного ускорения, так же, как и при сглаживании, используется кривая, которая может пригодиться в тех случаях, когда объекты (например, системные уведомления) вылетают за пределы экрана на полной скорости. Такой характер движения подходит только для тех случаев, когда объекты невозвратно пропадают с экрана.

Плавное замедление. Данная анимация является обратной для плавного ускорения, объект проходит большое расстояние и снижает скорость движения до полной остановки.

Ускорение-замедление или стандартная кривая. Эта анимация позволяет создать эффект плавного ускорения и плавного замедления объекту. Лучше использовать асимметричные кривые – так движение будет более естественным и реалистичным.

Хореография анимации в интерфейсе. Различают два подхода к хореографии в анимации элементов интерфейса: наличие ведущего объекта и равнозначность объектов.

Наличие ведущего объекта позволяет упорядочить содержимое экрана и расставить акценты. Ведущий (центральный) элемент будет привлекать внимание пользователя, при этом остальные элементы перейдут на задний план.

При равнозначности объектов их перемещение подчиняется единому правилу. Если не соблюдать порядок, то внимание пользователя рассеивается, а появление всех элементов сразу единовременно создаст большое количество точек фокуса.

Анимация в интерфейсе направлена на имитацию движения объектов в реальном мире – сопротивление, ускорение и т.п. При правильном построении анимации пользователь не отвлекается от

своих целей. Эффекты не должны замедлять пользователя и вставать на пути решения его задач.

Разработка интерфейса по макетам

По окончании работы над визуальной частью интерфейса осуществляется разработка интерфейса по макетам. Рассмотрим основные компоненты, которые необходимо передать разработчику.

- Дизайн-макет приложения, выполненный в Sketch, Figma, Photoshop, Illustrator;
- карта экранов приложения, выполненная в XMind, MS Visio или другом похожем по функционалу инструменте;
- кликабельный прототип, выполненный в Marvel, InVision или другом похожем по функционалу инструменте;
- иконки и иллюстрации, использованные в дизайн-макете для всех плотностей экранов;
- шрифты, использованные в дизайн-макете;
- видео-файлы со всеми нестандартными анимациями или ссылки на примеры таких анимаций из других приложений.

Правильное проектирование пользовательского интерфейса снижает количество ошибок пользователя, уменьшает затраты на техническую поддержку ИС, а также на изменение пользовательского интерфейса по требованию клиентов.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Компаниец, В.С. Проектирование и юзабилити – исследование пользовательских интерфейсов: уч.пособ. / В.С. Компаниец, А.Е. Лызь. – Издательство Южного федерального университета, 2020. – 107 с.

2. Малышев, К.В. Построение пользовательских интерфейсов / К.В. Малышев. — Москва: ДМК Пресс, 2021. – 268 с.

3. Терещенко, П.В. Интерфейсы информационных систем: учебное пособие / П.В. Терещенко, В.А. Астапчук. – Новосибирск: НГТУ, 2012. – 67 с.

УДК 004.891

Д.В. ШАРОНОВ, М.А. ИВАНЧИКОВА
Рязанский государственный радиотехнический университет
имени В.Ф. Уткина

РЕАЛИЗАЦИЯ НЕЙРОННОЙ СЕТИ ДОЛГОЙ КРАТКОСРОЧНОЙ ПАМЯТИ СРЕДСТВАМИ .NET

Рассматривается процесс реализации нейронной сети долгой краткосрочной памяти средствами платформы .NET.

Ячейка нейросети долговременной краткосрочной памяти (LSTM) – это небольшой программный компонент, который можно использовать для создания рекуррентной нейронной сети, способной делать прогнозы относительно последовательностей данных. Сети LSTM ответственны за крупные прорывы в нескольких областях машинного обучения. В данной статье будет описан процесс реализации сети LSTM с помощью C#. Это позволит точнее понять работу ячеек LSTM, что будет полезно при создании нейронной сети LSTM с использованием библиотек, таких как Microsoft CNTK или Google TensorFlow.

Описание ячейки LSTM

Архитектура ячейки представлена на рисунке 1.

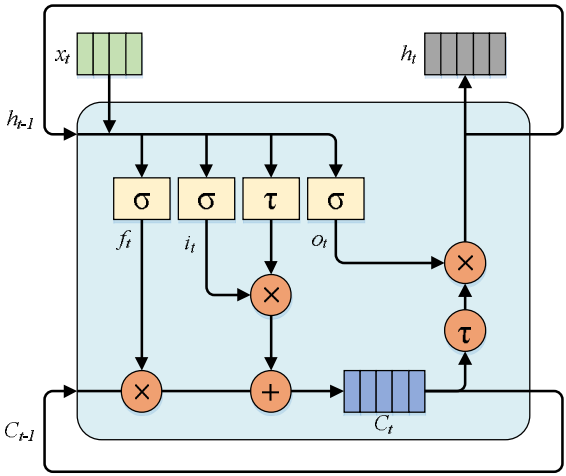


Рисунок 1 – Архитектура ячейки LSTM

На вход ячейки поступают входной вектор x_t размерности $n \times 1$, выходной вектор с предыдущего слоя h_{t-1} размерности $m \times 1$ и вектор состояния с предыдущего слоя C_{t-1} размерности $m \times 1$.

На выход поступают выходной вектор h_t размерности $m \times 1$ и новый вектор состояния C_t размерности $m \times 1$.

Вектор состояния является ключевой особенностью сетей LSTM, позволяющей запоминать информацию с предыдущих итераций работы нейронной сети.

Поведение ячейки LSTM определяется математическими уравнениями:

$$f_t = \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right) \quad (1)$$

$$i_t = \sigma\left(W_i x_t + U_i h_{t-1} + b_i\right) \quad (2)$$

$$o_t = \sigma\left(W_o x_t + U_o h_{t-1} + b_o\right) \quad (3)$$

$$C_t = f_t \times C_{t-1} + i_t \times \tau\left(W_c x_t + U_c h_{t-1} + b_c\right) \quad (4)$$

$$h_t = o_t \times \tau\left(C_t\right) \quad (5)$$

Уравнения (1), (2) и (3) определяют три вентиля: вентиль забывания, вентиль ввода и вентиль вывода. Каждый вентиль представляет собой вектор значений от 0,0 до 1,0, которые используются для определения того, сколько информации следует забыть (запомнить) в каждом цикле ввода-вывода. Уравнение (4) вычисляет новое состояние ячейки, а уравнение (5) – новый выходной вектор.

Реализация нейронной сети

Реализуемая нейросеть представлена классами:

- Network.cs – нейронная сеть LSTM;
- Module.cs – модуль (ячейка) LSTM;
- Layer.cs – абстрактный класс слоя ячейки нейронной сети;
- SigmaLayer.cs – сигмоидальный слой ячейки нейронной сети;
- TanhLayer.cs – слой ячейки нейронной сети с функцией активации гиперболического тангенса.

Диаграмма классов представлена на рисунке 2.

При использовании нейросети ей передаётся входной вектор, размерность входного вектора n, размерность выходного вектора и вектора состояния m и количество итераций работы нейронной сети.

Последовательно создаются модули LSTM. В первый модуль передаются нулевые выходной вектор и вектор состояния. Входными данными для последующих модулей являются выходные данные предыдущего.

В каждой ячейке LSTM производятся математические преобразования векторов с использованием слоёв нейронной сети, в соответствии с её архитектурой.

Каждый слой ячейки содержит матрицу входных весов W размерности $m \times n$, матрицу выходных весов U размерности $m \times m$ и сдвиг размерности $m \times 1$. Соответствующие веса и сдвиг определяются при обучении нейронной сети и хранятся в xml-файлах.

В каждом слое матрица входных весов умножается на входной вектор, матрица выходных весов умножается на выходной вектор предыдущего слоя, и к сумме произведений прибавляется

соответствующий сдвиг. После каждый элемент вектора преобразовывается соответствующей активационной функцией.

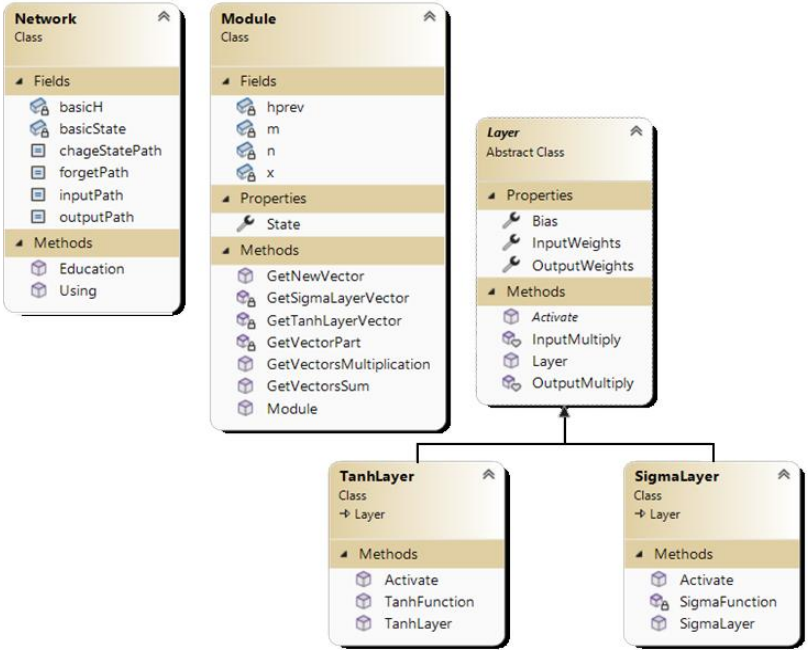


Рисунок 2 – Диаграмма классов

Результатом работы нейронной сети является список предсказанных векторов.

При обучении нейронной сети на каждой итерации работы нейронной сети ей так же передаётся правильный вектор, который она должна предсказать. Веса слоёв нейронной сети корректируются в зависимости от текущих ошибок предсказания.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. McCaffrey J. Understanding LSTM Cells Using C#. – Текст: электронный. – 2018. – URL: https://learn.microsoft.com/en-us/archive/msdn-magazine/2018/april/test-run-understanding-lstm-cells-using-csharp/ (дата обращения: 15.04.2023).

2. Olah Ch. Understanding LSTM Networks. – Текст: электронный. – URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/ (дата обращения: 15.04.2023).

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В ПРИКЛАДНЫХ ИССЛЕДОВАНИЯХ

Межвузовский сборник научных трудов

Компьютерная верстка: А.Н. Сапрыкин, А.О. Сапрыкина
Изображение обложки сборника взято с сайта freepik.com

Подписано в печать 12.05.2023.
Формат 60x84 1/16. Бумага офсетная.
Печать цифровая. Гарнитура Times.
Печ. л. 15,37. Тираж 50 экз. Заказ № 6121.

ISBN 978-5-907568-65-5

Издательство ИП Коняхин А.В. (Book Jet)

Отпечатано в типографии Book Jet
390005, г. Рязань, ул. Пушкина, д. 18
Сайт: http://bookjet.ru
Почта: info@ bookjet.ru
тел.:+7 (4912) 466-151