МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра программной инженерии

УТВЕРЖДАЮ
Проректор по учебной работе
О.Г. Локтионова
«15» 10 2024 г.

ЭКСПЕРТНЫЕ СИСТЕМЫ

Методические рекомендации по выполнению лабораторных работ для студентов направления подготовки 09.04.04 – Программная инженерия

Курск 2024

УДК 004
Составитель Т.Н. Конаныхина

Рецензент
к.т.н. Стародубцева Л.В.

Экспертные системы: методические рекомендации по выполнению лабораторных работ направления подготовки 09.04.04 – Программная инженерия / Юго-Зап. гос. ун-т; сост.: Т.Н. Конаныхина, Курск, 2024. 50 с.

Содержат методические рекомендации к выполнению лабораторных работ по дисциплине «Экспертные системы».

Методические указания по структуре, содержанию и стилю изложения материала соответствуют методическим и научным требования, предъявляемым к учебным и методическим пособиям.

Предназначены для студентов направления подготовки 09.04.04 – Программная инженерия.

Текст печатается в авторской редакции

Подписано в печать_____. Формат 60x84 1/16
Усо.печ.л.3,54. Уч.-изд.л.3,18.Тираж100экз. Заказ:1143Бесплатно.
Юго-Западный государственный университет.
305040. г. Курск, ул. 50 лет Октября, 94.

Инструкция по технике безопасности

1. Общие требования безопасности

1.1.К работе на персональном компьютере допускаются лица, прошедшие обучение безопасным методам труда, вводный инструктаж, первичный инструктаж на рабочем месте.

1.2.При эксплуатации персонального компьютера на человека могут оказывать действие следующие опасные и вредные производственные факторы:

- повышенный уровень электромагнитных излучений;
- повышенный уровень статического электричества;
- пониженная ионизация воздуха;
- статические физические перегрузки;
- перенапряжение зрительных анализаторов.

1.3. Студент при выполнении лабораторных работ обязан:

1.3.1.Выполнять только ту работу, которая определена в методических указаниях.

1.3.2.Содержать в чистоте рабочее место.

1.3.3.Соблюдать режим труда и отдыха.

1.3.3.Соблюдать меры пожарной безопасности.

1.4.Рабочие места с компьютерами должны размещаться таким образом, чтобы расстояние от экрана одного видеомонитора до тыла другого было не менее 2,0 м, а расстояние между боковыми поверхностями видеомониторов - не менее 1,2 м.

1.5.Рабочие места с персональными компьютерами по отношению к световым проемам должны располагаться так, чтобы естественный свет падал сбоку, преимущественно слева.

1.6.Оконные проемы в помещениях, где используются персональные компьютеры, должны быть оборудованы регулируемыми устройствами типа: жалюзи, занавесей, внешних козырьков и др.

1.7.Рабочая мебель для пользователей компьютерной техникой должна отвечать следующим требованиям:

- высота рабочей поверхности стола должна регулироваться в пределах 680 - 800 мм; при отсутствии такой возможности высота рабочей поверхности стола должна составлять 725 мм;
- рабочий стол должен иметь пространство для ног высотой не менее 600 мм, глубиной на уровне колен не менее 450 мм и на уровне вытянутых ног не менее 650 мм;

2. Требования безопасности перед началом работы

2.1.Подготовить рабочее место.

2.2.Отрегулировать освещение на рабочем месте, убедиться в отсутствии бликов на экране.

2.3.Проверить правильность подключения оборудования к электросети.

2.4.Проверить исправность проводов питания и отсутствие оголенных участков проводов.

2.5.Проверить правильность установки стола, стула, подставки для ног, пюпитра, угла наклона экрана, положение клавиатуры, положение "мыши" на специальном коврике, при необходимости произвести регулировку рабочего стола и кресла, а также расположение элементов компьютера в соответствии с требованиями эргономики и в целях исключения неудобных поз и длительных напряжений тела.

3. Требования безопасности во время работы

3.1.Студенту при работе на ПК запрещается:

- прикасаться к задней панели системного блока (процессора) при включенном питании;
- переключать разъемы интерфейсных кабелей периферийных устройств при включенном питании;
- допускать попадание влаги на поверхность системного блока (процессора), монитора, рабочую поверхность клавиатуры, дисководов, принтеров и других устройств;

- производить самостоятельное вскрытие и ремонт оборудования;
- работать на компьютере при снятых кожухах;
- отключать оборудование от электросети и выдергивать электровилку, держась за шнур.

3.2.Продолжительность непрерывной работы с компьютером без регламентированного перерыва не должна превышать 2-х часов.

3.3.Во время регламентированных перерывов с целью снижения нервно - эмоционального напряжения, утомления зрительного анализатора, устранения влияния гиподинамии и гипокинезии, предотвращения развития познотонического утомления выполнять комплексы упражнений.

4. Требования безопасности в аварийных ситуациях

4.1.Во всех случаях обрыва проводов питания, неисправности заземления и других повреждений, появления гари, немедленно отключить питание.

4.2.Не приступать к работе до устранения неисправностей.

4.3.При получении травм или внезапном заболевании немедленно известить организовать первую доврачебную помощь или вызвать скорую медицинскую помощь.

5. Требования безопасности по окончании работы

5.1.Отключить питание компьютера.

5.2.Привести в порядок рабочее место.

5.3.Выполнить упражнения для глаз и пальцев рук на расслабление.

Мини проект: Проектирование и создание экспертной системы

Цели создания мини-проекта

Развить навыки создания и применения экспертных систем на практических примерах, используя полученные знания в области программирования и системного анализа. Развить умения анализа требований, проектирования архитектуры системы и реализации программного обеспечения с учетом современных методов и технологий. Стимулировать анализ и критическую оценку различных подходов к решению задач и усвоение лучших практик в разработке программного обеспечения.

Задачи работы

Спроектировать структуру системы, включая выбор технологий и инструментов, необходимых для реализации проекта.

Собрать и разработать базу знаний для экспертной системы, включая правила, факты и алгоритмы вывода.

Кодировать основные компоненты системы, включая интерфейс пользователя, логику обработки и хранения данных.

Провести тестирование системы на предмет выявления ошибок и обеспечения качества работы, а также выработать методы для улучшения работы системы.

Создать техническую документацию, включая описание архитектуры, используемых технологий, алгоритмов и инструкций пользователя.

Презентация результатов: Подготовить и провести презентацию проекта, в которой студенты должны рассказать о своей работе, подходах, оправдании выбора решений и продемонстрировать функционирование системы.

Ожидаемые результаты

Создание работоспособной экспертной системы, соответствующей проанализированным требованиям. Развитие навыков программирования и проектного мышления. Улучшение навыков командного взаимодействия и управления проектами. Понимание принципов работы экспертных систем и их применения в различных областях.

Ход работы.

1. Изучите теоретический материал методических указаний для лабораторных работ и приложение с примерами кода методических указаний для самостоятельной работы студентов по дисциплине.
2. Определите цели проектирования ЭС и требования к её разработке.
3. Поиск и анализ данных.
4. Проектирование структуры базы данных и/или базы знаний.
5. Выбор инструментов и технологий.
6. Реализация проекта.
7. Тестирование системы.
8. Улучшение разработанной экспертной системы с помощью внедрения методов машинного обучения (продвинутый).

1. Этапы проектирования экспертной системы

Анализ и определение проблемы

Проектирование экспертной системы — это сложный многоэтапный процесс, целью которого является создание системы, способной принимать решения на уровне, сопоставимом с опытом квалифицированного эксперта в конкретной области. В этом разделе мы рассмотрим первый этап проектирования — анализ и определение проблемы — и разберем его на основные подпункты.

Анализ и определение проблемы — это критически важный этап в разработке экспертной системы, который закладывает основу для всего

проекта. Здесь необходимо точно сформулировать нужды пользователя, специфику задачи, которую должна решать система, и контекст, в котором она будет использоваться. Неправильное определение проблемы приведет к созданию системы, которая не удовлетворяет требованиям пользователей и не решает реальных задач.

Первым шагом в анализе проблемы является выявление потребностей пользователей. Это можно сделать через:

- Общение с конечными пользователями, экспертами в области, и заинтересованными сторонами помогает понять их ожидания и требования к системе.
- Анкеты позволяют собрать данные от более широкого круга пользователей и выявить общие проблемы или потребности.
- Изучение текущих процессов и методов, используемых в данной области, поможет определить, какие проблемы существующая система не может решить.

Анализ потребностей позволит избежать непонимания между разработчиками и пользователями и поможет определить, какие функции должны быть включены в систему.

Определение области применения системы — это следующий ключевой шаг. Здесь важно четко обозначить, какие задачи и проблемы будет решать экспертная система. В рамках этого подпункта необходимо:

- Уточнить конфигурацию задачи: Определить, является ли задача четко структурированной или неструктурированной. Это повлияет на выбор методов и подходов к проектированию системы.
- Сформулировать области применения: Ясно обозначить, в каких случаях и для каких пользователей система будет использована. Например, экспертные системы в медицине могут быть нацелены на диагностику, выбор лечения или поддержку принятия решений для врачей.

Определение области применения помогает избежать непоследовательного подхода в разработке, а также минимизирует риски, связанные с созданием ненужных функций.

Перед запуском разработки следует провести анализ существующих решений в аналогичной области. Это поможет выявить, какие проблемы были успешно решены другими системами, а также определить слабые места, которые можно улучшить. Этап анализа включает:

- Исследование литературы: Изучение научных публикаций, патентов и отчетов о работе существующих экспертных систем в данной области позволяет получить представление о текущих достижениях.
- Обзор коммерческих решений: Анализ доступных на рынке экспертных систем может дать идеи для функциональности и специфики разработки.

Также важно вовлечь экспертов в процессе анализа существующих решений, чтобы выявить области, которые требуют улучшения и уточнения в новых системах.

Формализация проблемы заключается в том, чтобы перевести выявленные потребности пользователей и специфику задачи в четкие и конкретные критерии, которые система должна удовлетворять. Этот этап включает:

- Определение критериев успеха: Установить метрики, по которым будет оцениваться эффективность работы системы. Это могут быть показатели точности, скорости принятия решений и удовлетворенности пользователей.
- Постановка задач: Записать конкретные задачи, которые система должна решать, например, диагностика болезни, прогнозирование финансовых тенденций или рекомендации по производственным процессам.

Формализация проблемы критична для эффективного управления проектом. Хорошо прописанные критерии помогают в дальнейшем процессе разработки, тестирования и оценки системы.

Каждый проект сталкивается с определенными ограничениями, такие как:

- Технические ограничения: Ограничения по аппаратному обеспечению, программному обеспечению и необходимым ресурсам.
- Финансовые ограничения: Бюджетные лимиты, которые могут повлиять на возможности разработки.
- Временные ограничения: Сроки реализации проекта, которые могут ограничить глубину анализа и функциональность системы.

Признание и документирование этих ограничений на начальном этапе позволяет избежать неприятных сюрпризов в дальнейшем, обеспечивая более реалистичный подход к проектированию системы.

После того как потребности и ограничения были определены, важно согласовать их с заинтересованными сторонами. Это включает:

- Проведение встреч: Презентация собранной информации пользователям, экспертам и производителям для обсуждения и уточнения требований.
- Документирование требований: Все требования и ограничения должны быть задокументированы и согласованы всеми сторонами, чтобы избежать недоразумений на следующих этапах.

Согласование требований гарантирует, что все участники проекта находятся на одной волне и что система будет разрабатываться с учетом всех заинтересованных мнений.

Важно помнить, что в процессе разработки экспертной системы могут вноситься изменения. Учитывая это, следует:

- Подготовить систему к адаптации: Изучить возможность обновления системы с учетом изменяющихся требований пользователей и технологического прогресса.
- Регулярно пересматривать требования: Создать механизм для пересмотра и актуализации требований на протяжении всего жизненного цикла проекта.

Поскольку области применения экспертных систем могут меняться, гибкость в планировании и анализе позволит поддерживать актуальность системы.

Этап анализа и определения проблемы — это важное основание для создания успешной экспертной системы. Внимательное исследование потребностей пользователей, определение области применения, анализ существующих решений и формализация задачи закладывают отправную точку для всего процесса разработки. Этот этап требует внимания, тщательности и вовлеченности всех заинтересованных сторон, что в конечном итоге приводит к созданию системы, способной решать реальные проблемы и соответствовать ожиданиям пользователей.

2. Сбор и представление знаний

После анализа и определения проблемы следующим критически важным этапом в проектировании экспертной системы является сбор и представление знаний. Это процесс, который включает в себя не только извлечение знаний от экспертов и из других источников, но и организацию этих знаний в структуру, которая будет понятна и доступна для обработки системой. На этом этапе важно учесть все аспекты, чтобы обеспечить точность и эффективность знаний, используемых в системе.

Сбор знаний — это процесс извлечения информации, которая необходима для принятия решений в конкретной области. Он включает в себя следующие подэтапы:

1. Взаимодействие с экспертами: Основная цель этого этапа — получить информацию от специалистов, которые обладают опытом и знаниями. Интервью, обсуждения и мозговые штурмы могут помочь выявить важные аспекты задачи.

2. Исследование документации: Сбор материалов, таких как учебные пособия, исследования, руководства и пр., также важен, поскольку в них можно найти структурированные и проверенные знания.

3. Анализ данных: Это может включать в себя обработку исторических данных и статистики, что позволит извлечь полезные паттерны и знания, которые могут быть полезны в процессе принятия решений.

4. Преобразование знаний: Важно не только собрать знания, но и преобразовать их в структуру, подходящую для дальнейшего анализа и использования.

Сбор знаний можно осуществлять с помощью различных методов, включая:

- Экспертные интервью: Получение информации непосредственно от профессионалов в области, что позволяет получить не только фактические данные, но и интуитивные подходы и мнения.
- Анкетирование: Средство для сбора данных от более широкой группы людей и аналитических экспертов, позволяющее формализовать ответы и сложно структурировать входящую информацию.
- Наблюдение: Прямое наблюдение за работой экспертов в реальных условиях позволяет понять нюансы процессов, которые могут быть не задокументированы.
- Групповая работа: Проведение семинаров и мозговых штурмов, где эксперты могут сотрудничать и обмениваться идеями, помогает собрать более разнообразные знания.

Процесс представления знаний включает в себя структурирование и организацию собранных данных в логическую и понятную форму. Эффективное представление знаний существенно влияет на работу

экспертной системы. Существует несколько широко используемых форматов для представления знаний.

Это одна из самых распространенных форм представления знаний в экспертных системах. Правила заданы в формате "если-то" (IF-THEN), что позволяет системе легко интерпретировать, применять и выводить новые знания. Например:

- IF температура > 37.5 THEN возможна инфекция.

Такой формат позволяет системе быстро принимать решения на основе набора правил.

Семантические сети представляют знания в виде графа, где узлы обозначают концепции или объекты, а ребра отражают отношения между этими концепциями. Это позволяет визуализировать взаимосвязи между различными аспектами области знаний и может быть крайне полезным для сложных систем.

Деревья решений представляют собой графическую модель, позволяющую пользователю следовать логике принятия решения. Каждый узел дерева представляет собой вопрос или условие, а ветви показывают ответ. Это делает систему интуитивно понятной и доступной для пользователей, так как они могут следить за процессом принятия решения.

Создание онтологий — это более сложный и продвинутый подход к представлению знаний. Онтология формализует концепции и их взаимосвязи, обеспечивая четкое понимание области. Она может содержать:

- Классы: Основные категории, которые описывают сущности.
- Атрибуты: Характеристики классов, которые используются для описания объектов.
- Отношения: Указывают, как классы взаимосвязаны между собой.

Онтологии позволяют более точно вести учет знания и упрощают взаимодействие между системой и пользователями.

Качество собранных и представленных знаний критически важно для успешной работы экспертной системы. Основные критерии качества включают:

- Точность: Знания должны быть максимально точными и основанными на проверенных данных. Неверная информация может привести к ошибочным выводам.
- Полнота: Важно, чтобы система охватывала все аспекты и нюансы рассматриваемой области. Пробелы в знаниях могут ограничить способности системы.
- Актуальность: Знания должны быть актуальными, особенно в областях, где информация постоянно обновляется, такие как медицина или технологии.
- Согласованность: Знания не должны противоречить друг другу. Систематизация и проверка знаний помогут избежать этой проблемы.

После сбора и представления знаний они подлежат проверке и валидации. На этом этапе осуществляется:

- Тестирование правил: Проводится тестирование собранных правил и моделей на реальных или симулированных данных, чтобы проверить их эффективность и точность.
- Обратная связь от экспертов: Обсуждение представленных знаний с экспертами позволяет выявить пробелы и улучшить систему. Это также помогает удостовериться, что система отвечает реальным потребностям пользователей.
- Обновление базы знаний: Необходимо обеспечить регулярное обновление и пересмотр знаний по мере поступления новой информации или изменения условий, в которых работает система.

При сборе и представлении знаний могут использоваться различные инструменты и технологии. Это включает в себя:

- Системы управления базами данных (СУБД): Используются для хранения и управления большими объемами структурированных данных.

- Инструменты для визуализации данных: Позволяют представлять знания в графическом виде, что облегчает восприятие информации и выявление взаимосвязей.
- Программные средства для разработки онтологий: Такие как Protégé, которые могут помочь в создании и управлении онтологиями.
- Инструменты для обработки текстов: Используются для автоматизации сбора знаний из большого объема текстовой информации, такой как статьи или учебные материалы.

Рассмотрим пример экспертной системы в области медицины, которую разрабатывает команда с целью диагностики заболеваний. Процесс может выглядеть следующим образом:

1. Сбор знаний: Интервью с врачами, анализ медицинских справочников и диагнозов.
2. Представление знаний: Создание правил вывода, например, в формате IF-THEN для различных симптомов и их возможных заболеваний.
3. Валидация знаний: Тестирование системы с использованием реальных медицинских кейсов и получение обратной связи от пользователей.
4. Обновление знаний: Регулярное пересмотр знаний с учетом новых исследований и клинических рекомендаций.

Сбор и представление знаний являются основополагающими этапами в разработке экспертных систем. Эффективное осуществление этих этапов требует внимательного подхода к взаимодействию с экспертами, тщательного анализа информации и четкой структуризации знаний. Правильное представление информации в нужной форме обеспечивает систему необходимыми данными для принятия решений и выполнения задач, что в итоге определяет успешность всего проекта.

3. Реализация и тестирование системы

После того как знания были собраны и представленные в удобной для обработки форме, команда разработки переходит к этапу реализации и

тестирования экспертной системы. Это один из самых критических этапов, на котором осуществляется создание программного обеспечения, интеграция компонентов и проверка системы на соответствие заданным требованиям. Это включает в себя несколько ключевых подпунктов, каждый из которых имеет свои особенности и важность.

1. Процесс реализации системы

Реализация системы представляет собой процесс преобразования проектных спецификаций и прототипирования в работающую программу. На этом этапе разработчики должны обратить внимание на множество аспектов:

- Выбор технологий: Определение языков программирования, платформ и инструментов разработки, которые будут использоваться для создания экспертной системы. Часто выбираются языки, которые поддерживают разработку с использованием логики и обработки данных, такие как Python, Prolog или Java.
- Архитектура системы: Формирование архитектуры системы, которая определяет, как различные компоненты будут взаимодействовать. Это может быть компонентная архитектура, где система состоит из отдельных модулей, или монолитная архитектура, где вся логика связана в одном приложении.
- Реализация базы знаний: Программирование загрузки и обработки базы знаний, которая была собрана и представлена ранее. Это включает в себя разработку правил вывода, механизмов работы с данными и логики применения знаний.
- Интерфейс пользователя: Создание интуитивного и удобного пользовательского интерфейса, который позволит пользователю взаимодействовать с системой. Это может включать графические интерфейсы (GUI), командные интерфейсы или веб-приложения. User Experience (UX) играет ключевую роль на этом этапе.
- Интеграция компонентов: Включает в себя интеграцию базы данных, средств обработки данных и пользовательского интерфейса в единое

целое. Работа каждого компонента должна быть согласована и обеспечивать бесперебойное взаимодействие.

2. Тестирование системы

Тестирование является важным этапом, который позволяет выявить ошибки и проблемы в работе системы до ее внедрения. Оно включает в себя несколько уровней тестирования:

- Проверка компонентов: На этом начальном уровне тестируется каждая часть системы по отдельности. Например, тестируются правила вывода, функции взаимодействия с базой данных и элементы интерфейса. Это позволяет выявить ошибки на раннем этапе и упрощает процесс отладки.
- Интеграционное тестирование: На этом уровне тестируются взаимодействия между компонентами системы. Это важно для того, чтобы убедиться, что различные модули корректно работают вместе и не вызывают сбоев при взаимодействии.
- Системное тестирование: На этой стадии проверяется вся система как единое целое. Тестируются реальные сценарии использования системы с использованием тестовых данных, чтобы убедиться, что система выполняет свои функции в соответствии с требованиями.
- Тестирование на соответствие требованиям: Проверка, соответствует ли система всем требованиям, установленным на предыдущих этапах проектирования. Это включает в себя как функциональные, так и нефункциональные требования (например, производительность, безопасность и удобство использования).
- Тестирование на устойчивость: Испытания системы под нагрузкой позволяют проверить, как она будет реагировать на стрессовые ситуации, такие как высокая загрузка пользователей или большие объемы данных. Это необходимо для оценки ее надежности в реальных условиях.

3. Валидация и верификация

На этапе тестирования важно провести валидацию и верификацию системы:

- Верификация: Это процесс проверки того, что система была построена правильно согласно спецификациям и проекту. Другими словами, верификация отвечает на вопрос: "Мы построили систему правильно?"
- Валидация: Этот процесс фокусируется на том, удовлетворяет ли система потребностям пользователей и решает ли поставленные задачи. Валидация отвечает на вопрос: "Мы построили нужную систему?"

Чтобы провести эти процессы, часто используются различные техники, включая тестирование с использованием реальных пользователей и сбор обратной связи от экспертов.

4. Обратная связь и доработка

После завершения тестирования крайне важно собрать обратную связь. Это можно сделать через:

- Пользовательские тестирования: Привлечение конечных пользователей системы для проведения тестов и оценка их взаимодействия с системой. Как они воспринимают интерфейс? Удобно ли им использовать систему? Это все поможет выявить недостатки или области для улучшения.
- Анализ ошибок: Изучение всех выявленных ошибок и их причин для внесения необходимых изменений в проект. Ошибки могут быть как критическими, так и незначительными, и каждый из них нужно проанализировать, чтобы избежать их повторения.
- Отзывы экспертов: Обсуждение системы с экспертами, которые могут дать ценные рекомендации по улучшению функциональности и качества. Эксперты могут обнаружить проблемы, которые пользователи, возможно, не заметили.

5. Документация

Документация является важной частью реализации и тестирования. На этом этапе создается:

- Пользовательская документация: Инструкции и руководства для пользователей, которые помогут им эффективно взаимодействовать с

системой. Это может включать в себя пошаговые гайды, FAQ и технические условия.

- Документация для разработчиков: Описание архитектуры системы, используемых технологий, а также процесс интеграции компонентов. Это поможет будущим разработчикам и командам технической поддержки в обслуживании и улучшении системы.
- Отчеты о тестировании: Документирование всех проведенных тестов, выявленных проблем и их решений. Эти отчеты помогут в дальнейшем анализе и валидации системы.

6. Подготовка к внедрению

После завершения всех этапов реализации и тестирования настает время подготовки к внедрению системы в реальную эксплуатацию. В этот момент стоит учесть:

- Обучение пользователей: Проведение тренингов для конечных пользователей, чтобы они могли максимально эффективно использовать систему. Это может включать как практические занятия, так и теоретические пояснения.
- Планирование внедрения: Создание плана внедрения системы, включая временные рамки, этапы, ответственных лиц и ресурсы, необходимые для успешного запуска.
- Пост-внедренческая поддержка: Разработка стратегии технической поддержки после внедрения системы. Пользователи могут столкнуться с проблемами, которые потребуют быстрой реакции со стороны команды разработчиков.

Пример успешной реализации и тестирования

Рассмотрим пример экспертной системы, разработанной для диагностики заболеваний. Процесс реализации может выглядеть так:

1. Реализация: Команда разработала систему на Python, используя библиотеку для работы с данными. База знаний была интегрирована с логикой вывода по правилам "IF-THEN".

2. Тестирование: Каждый компонент системы был протестирован отдельно, затем проведено интеграционное тестирование. Были разработаны реальные сценарии для системного тестирования, включая верификацию на соответствие медицинским стандартам.

3. Обратная связь: Конечные пользователи, включая врачей, были вовлечены в тестирование системы, и их отзывы помогли улучшить интерфейс и функциональность.

4. Документация: Подготовлен полный набор документов, включая руководства для врачей и технические спецификации для IT-поддержки.

Этап реализации и тестирования является критически важным в разработке экспертных систем. Он обеспечивает функциональность и устойчивость системы, а также готовит ее к успешной эксплуатации в реальных условиях. Необходимость тщательной проверки и валидации системы делает этот этап основополагающим для достижения целей проекта и удовлетворения потребностей пользователей. Внимание к деталям на этом этапе закладывает основу для успешной работы экспертной системы в будущем.

4. Выбор инструментов и технологий

Языки программирования

В последние годы разработка экспертных систем значительно эволюционировала, и выбор языков программирования для их создания стал гораздо более разнообразным. Ранее популярные LISP и Prolog по-прежнему имеют свое место, но современный рынок предлагает множество новых инструментов и технологий. В этой секции мы рассмотрим основные языки программирования, которые активно используются для создания экспертных систем, таких как Python, Java, R и Julia.

1. Python

1.1. Общее представление

Python остается одним из наиболее популярных языков для разработки экспертных систем в последние годы. Он отличается своей простотой, богатой экосистемой библиотек и активным сообществом, что делает его идеальным выбором для исследователей и разработчиков.

1.2. Особенности Python

- Читаемость и простота: Синтаксис Python интуитивно понятен, что облегчает его изучение и использование, особенно для новых разработчиков.
- Богатая экосистема библиотек: С библиотеками, такими как TensorFlow, scikit-learn, NumPy и Pandas, Python предоставляет мощные инструменты для анализа данных и разработки моделей машинного обучения, что особенно важно для экспертных систем.
- Поддержка объектно-ориентированного программирования: Python позволяет организовывать код в классы и объекты, что улучшает структуру и повторное использование кода.

1.3. Применение Python в экспертных системах

Python нашел широкое применение в разработке систем, использующих машинное обучение, обработки естественного языка (NLP) и построения баз знаний. Например, современные экспертные системы для медицинской диагностики могут быть основаны на моделях, обученных с использованием библиотек машинного обучения Python.

2. Java

2.1. Общее представление

Java остается широко используемым языком программирования в бизнес-приложениях и системах. Его надежность, безопасность и поддержка многопоточного программирования делают его привлекательным для разработки сложных экспертных систем.

2.2. Особенности Java

- Портативность и кросс-платформенность: Java-Программы можно запускать на любой платформе, поддерживающей Java Virtual Machine (JVM), что делает его универсальным решением.
- Объектно-ориентированный подход: Java подходит для создания масштабируемых и структурированных систем за счет использования объектно-ориентированных принципов.
- Надёжность и безопасность: Java предлагает встроенные механизмы управления памятью и обработки исключений, что повышает надежность разрабатываемых систем.

2.3. Применение Java в экспертных системах

Java активно используется для разработки корпоративных экспертных систем, обеспечивающих помощь в принятии решений на основе анализа больших объемов данных. Например, в финансовых и медицинских секторах Java находит применение в системах, способных анализировать риски и предлагать рекомендации.

3. R

3.1. Общее представление

R — это язык программирования и среда для статистических вычислений и графики. Он стал популярным в научных исследованиях и в качестве инструмента для анализа данных, что делает его полезным для разработки экспертных систем, основанных на статистике.

3.2. Особенности R

- Инструменты для анализа данных: R предоставляет обширные пакеты для статистического анализа и визуализации данных, такие как ggplot2 и dplyr.
- Поддержка машинного обучения: Пакеты, такие как caret и randomForest, позволяют легко реализовывать и тестировать модели машинного обучения.

- Фокус на статистику: R идеально подходит для разработки систем, в которых необходимо проводить глубокий статистический анализ и делать выводы на основе данных.

3.3. Применение R в экспертных системах

R часто используется в медицинских и научных исследованиях для разработки экспертных систем, которые поддерживают принятие решений на основе статистических выводов. Например, системы, анализирующие клинические данные для прогнозирования исходов лечения.

4. Julia

4.1. Общее представление

Julia — это относительно новый язык программирования, который быстро завоевал популярность благодаря своей высокой производительности и возможностям для технических вычислений. Julia объединяет в себе простоту Python и производительность C, что делает его привлекательным для разработки экспертных систем.

4.2. Особенности Julia

- Высокая производительность: Julia разработан для выполнения высокопроизводительных вычислений, что делает его идеальным для обработки больших объемов данных.
- Читаемость и выраженность: Синтаксис Julia удобен и понятен, что облегчает разработку и понимание кода.
- Широкий набор библиотек: Julia обладает растущей экосистемой пакетов для статистики и машинного обучения, таких как Flux.jl и MLJ.jl.

4.3. Применение Julia в экспертных системах

Julia используется в научных исследованиях и финансовых приложениях, где необходима высокая скорость обработки данных. Экспертные системы, использующие Julia, могут эффективно анализировать данные и предоставлять точные рекомендации в реальном времени.

Таблица сравнения языков программирования, используемых для создания экспертных систем, с акцентом на удобство их применения:

Язык	Читабельность	Простота разработки	Поддержка библиотек	Производительность	Подходящие сферы применения	Интерпретируемый или компилируемый	Сообщество и поддержка	Портативность
Python	Высокая	Очень высокая	Огромная	Средняя	Обработка данных, машинное обучение, NLP	Интерпретируемый	Очень активное	Да
Java	Средняя	Высокая	Хорошая	Высокая	Корпоративные системы, большие приложения	Компилируемый	Активное	Да
R	Средняя	Высокая	Отличная	Низкая	Статистический анализ, научные исследования	Интерпретируемый	Активное	Да
Julia	Высокая	Высокая	Растущая	Очень высокая	Высокопроизводительные вычисления, АИ	Компилируемый	Растущее	Да

5. Платформы и фреймворки для разработки

Последние годы развитие технологий искусственного интеллекта и предиктивной аналитики значительно изменило подходы к созданию экспертных систем. Для разработки таких систем важно выбрать подходящую платформу или фреймворк. Эти инструменты помогают упростить процесс разработки, обеспечивая мощные возможности для интеграции, масштабирования и управления данными. В данной секции мы рассмотрим наиболее популярные платформы и фреймворки для разработки экспертных систем, которые стали востребованными за последние пять лет.

Основные платформы и фреймворки

1. TensorFlow

1.1. Общая информация

TensorFlow — это популярная открытая библиотека, разработанная Google, предназначенная для создания и обучения моделей машинного обучения и глубокого обучения. Платформа быстро завоевала популярность среди исследователей и разработчиков благодаря своей гибкости и масштабируемости.

1.2. Особенности TensorFlow

- Модульная архитектура: TensorFlow предлагает модульную архитектуру, что позволяет пользователям создавать сложные нейронные сети из простых компонентов.
- Поддержка различных языков программирования: Хотя основным языком программирования является Python, TensorFlow также поддерживает JavaScript, C++ и Java, что делает его более универсальным.
- Инструменты для визуализации: Благодаря TensorBoard разработчики могут визуализировать процесс обучения, что помогает в отладке и настройке моделей.

- Поддержка распределенных вычислений: TensorFlow позволяет обучать модели на нескольких устройствах, что значительно ускоряет процесс обучения.

1.3. Применение в экспертных системах

TensorFlow часто используется для создания экспертных систем, которые требуют сложного анализа данных, таких как системы предсказания, распознавание изображений, обработка естественного языка и многое другое. Возможности работы с большими объемами данных и гибкость архитектуры делают его идеальным выбором для модуля машинного обучения в экспертных системах.

2. PyTorch

2.1. Общая информация

PyTorch — это библиотека для машинного обучения, разработанная Facebook. Она завоевала популярность среди исследователей благодаря своей простоте в использовании и мощным возможностям для создания динамических вычислительных графов.

2.2. Особенности PyTorch

- Динамическое создание вычислительных графов: PyTorch позволяет создавать вычислительные графы "на лету", что делает написание и отладку кода более интуитивным.
- Гибкость моделей: Разработчики могут легко изменять архитектуру нейронных сетей в процессе обучения, что позволяет проводить эксперименты и оптимизировать модели.
- Поддержка GPU: PyTorch имеет отличную поддержку GPU, что позволяет значительно ускорить обучение моделей.
- Сообщество и ресурсы: Сообщество разработчиков активно поддерживает PyTorch, предоставляя множество учебных материалов и руководств.

2.3. Применение в экспертных системах

PyTorch широко используется для исследований в области глубокого обучения и создания экспертных систем, особенно в таких областях, как обработка естественного языка, компьютерное зрение и аудиовизуальный анализ. Его простота и гибкость позволяют быстро разрабатывать и внедрять прототипы.

3. Scikit-learn

3.1. Общая информация

Scikit-learn — это библиотека на Python для машинного обучения, предоставляющая простые и эффективные инструменты для анализа данных и построения моделей. Она идеально подходит для решения задач, связанных с обучением на основе примеров (supervised learning) и обучением без учителя (unsupervised learning).

3.2. Особенности Scikit-learn

- Скорость разработки: Scikit-learn предлагает интуитивно понятный интерфейс с простыми функциями для обучения, оценки и предсказания, что позволяет быстро создавать модели.
- Предобработанные данные: Библиотека включает множество инструментов для предобработки данных, включая нормализацию, очистку и отбор признаков.
- Модели и алгоритмы: Scikit-learn поддерживает множество алгоритмов машинного обучения, включая линейные модели, деревья решений, методы ансамблей и кластеризацию.
- Интеграция: Легко интегрируется с другими библиотеками Python, такими как NumPy и Pandas, что упрощает работу с данными.

3.3. Применение в экспертных системах

Scikit-learn идеально подходит для разработки экспертных систем, где необходима быстрая реализация и тестирование алгоритмов машинного обучения. Его функционал позволяет создавать модели, которые облегчают принятие решений на основе анализа данных, что делает его ценным инструментом в расширении возможностей экспертной системы.

Примеры и практические задания

Малая экспертная система, предназначенная для диагностики заболеваний на основе введенных симптомов

В последние пять лет наблюдается рост интереса к разработке экспертных систем, способных решать реальные проблемы в различных областях.

В этой лабораторной работе мы будем разрабатывать малую экспертную систему, которая будет предназначена для диагностики заболеваний на основе введенных симптомов. Эта система будет предназначена помочь пользователям получить предварительную оценку своего состояния и рекомендации о том, к какому специалисту им следует обратиться. Выберем язык программирования Python, так как он широко используется для разработки подобных приложений и обладает множеством библиотек для работы с данными и машинным обучением.

Шаг 1: Определение цели и требований

Перед началом разработки необходимо четко определить цель и задачи системы. Цель нашей экспертной системы заключается в том, чтобы позволить пользователям вводить свои симптомы и получать информацию о возможных заболеваниях и их лечении.

Основные функции системы:

1. Ввод данных о симптомах пользователем.
2. Сопоставление введенных данных с базой знаний о симптомах и заболеваниях.
3. Генерация списка возможных заболеваний и рекомендаций по дальнейшим действиям.

Требования к системе:

1. Простой и понятный интерфейс для пользователей.

2. Возможность работы с базой данных симптомов и заболеваний.

3. Возможность добавления новых данных в базу знаний.

Шаг 2: Поиск и анализ данных

Для создания базы знаний нам необходимо собрать данные о симптомах, заболеваниях и рекомендациях. Я воспользуюсь открытыми медицинскими ресурсами, такими как:

- **MedlinePlus**: предоставляет информацию о различных заболеваниях и их симптомах.
- **Wikipedia**: содержит обширную информацию о различных медицинских состояниях.
- **Медицинские справочники**: такие как "Клиническая диагностика" и т.д.

Основные категории данных:

1. Симптомы (например, головная боль, лихорадка).
2. Заболевания (например, грипп, мигрень).
3. Рекомендации (например, обратиться к врачу, пройти анализы).

Создадим таблицу с минимальным набором данных для тестирования – около 10-15 симптомов и 5-7 заболеваний.

Таблица симптомов

ID	Симптом	Описание
1	Головная боль	Ощущение боли в области головы.
2	Лихорадка	Повышенная температура тела.
3	Кашель	Рефлекторный акт очистки дыхательных путей.
4	Усталость	Ощущение физической или умственной усталости.
5	Тошнота	Неприятные ощущения в желудке, предшествующие рвоте.
6	Боль в горле	Неприятные ощущения или боль в горле при глотании.
7	Закладка носа	Усложненное дыхание из-за воспаления слизистой носа.
8	Сыпь	Ненормальная реакция кожи, проявляющаяся в виде

ID	Симптом	Описание
		покраснения или высыпаний.
9	Боль в животе	Ощущение боли в области живота.
10	Одышка	Затрудненное дыхание или чувство нехватки воздуха.

Таблица заболеваний

ID	Заболевание	Описание
1	Грипп	Вирусное инфекционное заболевание, проявляющееся лихорадкой, кашлем и ломотой в теле.
2	Мигрень	Неврологическое состояние, вызывающее сильные головные боли.
3	Аллергия	Реакция иммунной системы на определенные вещества, проявляющаяся сыпью и зудом.
4	ОРВИ	Острые респираторные вирусные инфекции, имеющие симптомы, такие как кашель и заложенность носа.
5	Пневмония	Воспаление легких, проявляющееся кашлем, одышкой и лихорадкой.

Таблица связи симптомов и заболеваний

В этой таблице можно указать, какие симптомы связаны с какими заболеваниями.

Disease ID	Symptom ID
1	2
1	3
1	4
1	5
2	1
3	9
4	3
4	7
5	2
5	3
5	10

Эти таблицы могут быть использованы для первичного тестирования экспертной системы, и в дальнейшем их можно будет расширять.

Шаг 3: Проектирование структуры базы данных

Для хранения данных мы используем реляционную базу данных. В Python можно управлять базами данных с помощью SQLite.

Структура базы данных:

1. **Таблица Symptom**:
    - id (INTEGER, PRIMARY KEY)
    - name (TEXT) — название симптома
2. **Таблица Disease**:
    - id (INTEGER, PRIMARY KEY)
    - name (TEXT) — название заболевания
3. **Таблица DiseaseSymptoms**:
    - disease_id (INTEGER, FOREIGN KEY) — ссылка на таблицу Disease
    - symptom_id (INTEGER, FOREIGN KEY) — ссылка на таблицу Symptom

Эта структура позволит сохранять много к одному соотношению между симптомами и заболеваниями.

Шаг 4: Выбор инструментов и технологий

В качестве инструмента для разработки я выбрал следующий стек технологий:

- **Язык программирования**: Python
- **Библиотеки**:
    - Flask: для создания веб-интерфейса
    - SQLite: для хранения данных
    - SQLAlchemy: для взаимодействия с базой данных

Создам виртуальное окружение для проекта с помощью **venv**, чтобы избежать конфликтов с другими проектами.

```bash
mkdir expert_system
cd expert_system
python -m venv venv
source venv/bin/activate  # Для Windows используйте venv\Scripts\activate
pip install Flask SQLAlchemy

```

Шаг 5: Реализация структуры базы данных

Теперь, когда у нас есть определённая структура, давайте создадим базу данных и наполним её начальными данными. Я создам файл **models.py**, в котором определю модели.

```python
from flask_sqlalchemy import SQLAlchemy

db = SQLAlchemy()

class Symptom(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(50), nullable=False)

class Disease(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(50), nullable=False)

class DiseaseSymptoms(db.Model):
    disease_id = db.Column(db.Integer, db.ForeignKey('disease.id'), primary_key=True)
    symptom_id = db.Column(db.Integer, db.ForeignKey('symptom.id'), primary_key=True)

```

Создание базы данных

Создадим файл **app.py**, в котором настроим приложение и создадим базы данных:

```python
from flask import Flask
from models import db, Symptom, Disease, DiseaseSymptoms

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///expert_system.db'
db.init_app(app)

with app.app_context():
    db.create_all()

```

Запускаем **app.py**, чтобы создать файл базы данных **expert_system.db**.

Шаг 6: Наполнение базы данных начальными данными

Теперь наполним таблицы **Symptom** и **Disease** начальными данными. Можно создать скрипт **populate_db.py** для этого:

```python
from app import app, db
from models import Symptom, Disease

with app.app_context():
    # Создание симптомов
    headache = Symptom(name='Головная боль')
    fever = Symptom(name='Лихорадка')
    cough = Symptom(name='Кашель')

    # Создание заболеваний
    flu = Disease(name='Грипп')
    migraine = Disease(name='Мигрень')

    # Добавление данных в базу
    db.session.add(headache)
    db.session.add(fever)
    db.session.add(cough)
    db.session.add(flu)
    db.session.add(migraine)
    db.session.commit()

```

Запускаем скрипт, чтобы наполнить базу данных.

Шаг 7: Реализация логики сопоставления симптомов и заболеваний

Теперь мы можем приступить к реализации логики, которая будет сопоставлять введенные симптомы с заболеваниями. Ниже показан пример функции, которая делает это:

```python
def diagnose(symptoms):
    results = []
    for symptom in symptoms:
        disease_symptoms = DiseaseSymptoms.query.filter_by(symptom_id=symptom.id).all()
        for ds in disease_symptoms:
            disease = Disease.query.get(ds.disease_id)
            if disease not in results:
                results.append(disease)
    return results

```

Эта функция принимает список симптомов и возвращает все заболевания, связанные с ними.

Шаг 8: Разработка пользовательского интерфейса

Теперь мы создадим простой веб-интерфейс на Flask. Я создам шаблоны для взаимодействия с пользователем.

Шаг 8.1: Настройка Flask

В файле **app.py** добавим маршруты для обработки запросов.

```python
from flask import render_template, request, redirect, url_for

@app.route('/')
def index():
    symptoms = Symptom.query.all()
    return render_template('index.html', symptoms=symptoms)

@app.route('/diagnose', methods=['POST'])
def diagnose_route():
    selected_symptoms = request.form.getlist('symptoms')
    symptoms_objects = [Symptom.query.get(int(s)) for s in selected_symptoms]
    diseases = diagnose(symptoms_objects)
    return render_template('result.html', diseases=diseases)

```

Шаг 8.2: Создание шаблонов

Создайте папку **templates**, а затем два файла: **index.html** и **result.html**.

index.html:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Экспертная система диагностики</title>
</head>
<body>
    <h1>Введите ваши симптомы</h1>
    <form action="/diagnose" method="post">
        {% for symptom in symptoms %}
            <input type="checkbox" name="symptoms" value="{{ symptom.id }}">{{ symptom.name }}<br>
        {% endfor %}
        <input type="submit" value="Диагностировать">
    </form>
</body>
</html>

```

result.html:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Результаты диагностики</title>
</head>
<body>
    <h1>Возможные заболевания</h1>
    <ul>
        {% for disease in diseases %}
            <li>{{ disease.name }}</li>
        {% endfor %}
    </ul>
    <a href="/">Назад</a>
</body>
</html>

```

Шаг 9: Тестирование системы

После завершения разработки система готова к тестированию. Я протестировал все основные функции, включая:

1. Ввод симптомов и правильное отображение данных.
2. Проверку диагностики на основе различных наборов симптомов.
3. Проверку на наличие ошибок в выводе.

Я собрал несколько запасных наборов данных, чтобы проверить систему с разными симптомами.

Шаг 10: Обратная связь и улучшения

После тестирования я планирую собрать обратную связь от своих однокурсников и преподавателей. Это позволит мне выявить недостатки и улучшить систему.

Возможные улучшения:

1. Расширение базы знаний: Добавление новых симптомов и заболеваний.
2. Улучшение интерфейса: Сделать его более удобным и современным.
3. Интеграция с API медицинских данных для получения актуальной информации.

Улучшение разработанной экспертной системы (продвинутый уровень)

Чтобы улучшить ранее описанную малую экспертную систему с использованием искусственного интеллекта, можно реализовать несколько этапов по внедрению методов машинного обучения и более интеллектуального подхода к диагностике. Вот несколько предложений по доработке системы.

Шаг 1: Сбор и расширение данных

1.1. Увеличение объемов данных

Исследуйте доступные открытые наборы данных о медицинских симптомах и заболеваниях. Варианты:

- **Kaggle**: Платформа с множеством наборов данных, включая медицинские.
- **PhysioNet**: Доступ к данным о медицинских состояниях.
- **OpenFDA**: Данные о лекарствах, заболеваниях и явлениях по медицинским исследованиям.

1.2. Сбор данных от пользователей

Рассмотрите возможность добавления функции для добровольного сбора данных от пользователей. Это может включать опросы или формы, чтобы пользователи могли сообщать о своих диагнозах и симптомах.

Шаг 2: Обработка и предобработка данных

2.1. Очистка данных

Убедитесь, что данные очищены от ошибок, пропущенных значений и неправильных форматов. Это критически важно для алгоритмов машинного обучения.

2.2. Кодировка категориальных переменных

Преобразуйте текстовые данные (например, названия симптомов и заболеваний) в числовые форматы. Это можно сделать с помощью методов:

- **Одинственной горячей кодировки (One-Hot Encoding)**.
- **Label Encoding**.

2.3. Нормализация

Если вы будете использовать численные показатели (например, возраст, время с появления симптомов), возможно применение нормализации для приведения данных к стандартному масштабу.

Шаг 3: Построение моделей машинного обучения

3.1. Выбор модели

Вы можете использовать несколько моделей для диагностики, такие как:

- **Наивный Байес**: Хорошо работает с категорическими переменными и помогает делать быстрые предположения.
- **Логистическая регрессия**: Полезна для бинарной классификации.
- **Деревья решений**: Позволяют лучше понять, какие симптомы ведут к какому заболеванию.
- **Сложные модели (например, Random Forest, XGBoost)**: Обеспечивают повышенную точность и могут обработать сложные взаимодействия между симптомами.

3.2. Обучение модели

Используйте данные для обучения модели. Например, создайте скрипт **train_model.py**, который будет собирать данные из базы и запускать обучение модели.

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Предполагаем, что у вас есть DataFrame с данными
data = pd.read_sql_query("SELECT * FROM your_data_table", connection)
X = data.drop('disease', axis=1)  # Удаляем колонку 'disease'
y = data['disease']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

model = RandomForestClassifier()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
print(f"Точность модели: {accuracy_score(y_test, y_pred)}%")

```

Шаг 4: Оценка и валидация модели

4.1. Оценка качества модели

Используйте метрики, такие как точность (accuracy), полнота (recall), и точность (precision), чтобы оценить вашу модель.

4.2. Кросс-валидацияПримените кросс-валидацию для более надежной оценки модели. Это поможет избежать переобучения (overfitting).

```python
from sklearn.model_selection import cross_val_score

scores = cross_val_score(model, X, y, cv=5)  # 5 фолд кросс-валидация
print(f"Точность кросс-валидации: {scores.mean()}")

```

Шаг 5: Интеграция модели в приложение

5.1. Создание предсказания

Добавьте функцию в ваше приложение, которая будет использовать обученную модель для предсказания заболеваний на основе введенных симптомов.

```python
@app.route('/predict', methods=['POST'])
def predict():
    symptoms = request.form.getlist('symptoms')
    # Преобразование введенных символов в нужный формат для модели
    input_data = preprocess(symptoms)  # Ваша функция обработки
    prediction = model.predict([input_data])  # Предсказание
    return render_template('result.html', diseases=prediction)

```

Если у вас есть достаточно данных, вы можете рассмотреть возможность применения более сложных моделей, таких как нейронные сети, которые могут помочь выявить скрытые зависимости между симптомами и заболеваниями.

Рассмотрите возможность использования методов NLP для обработки текстовых описаний симптомов, которые пользователи могут ввести, чтобы расширить возможности системы.

Интеграция искусственного интеллекта в экспертную систему не только повысит точность диагностики, но и сделает систему более адаптивной и эффективной. Эти изменения могут значительно улучшить взаимодействие пользователя с системой и обеспечить более качественные рекомендации по медицинским состояниям. Такой подход не только повысит доверие пользователей, но и упростит процесс диагностики и поиска необходимой информации.

Список предлагаемых тем для разработки экспертных систем (студент может предложить свою тему)

1. **Система диагностики домашних растений**: Разработка экспертной системы, которая помогает определить болезни растений и дает советы по уходу.
2. **Консультант по выбору профессии**: Система, которая анализирует интересы и навыки студентов и рекомендует подходящие профессии.

3. **Система поддержки принятия решений для управления финансами**: Программа, которая предлагает инвестиционные стратегии на основе анализа финансовых данных пользователя.

4. **Диагностика неисправностей автомобиля**: Экспертная система для определения возможных причин неисправностей на основе симптомов.

5. **Рекомендатор фильмов**: Система, которая предлагает фильмы по предпочтениям пользователя и их предыдущему опыту.

6. **Система оценки и выбора учебных курсов**: Анализ доступных курсов и рекомендация наилучших вариантов на основе целей студента.

7. **Автоматизация выбора медицинского обследования**: Система, которая помогает определить, какие медицинские тесты и обследования нужно пройти на основе симптомов.

8. **Экспертная система для планирования питания**: Программа, которая предлагает меню и рецепты на основе диетических предпочтений и ограничений пользователя.

9. **Система оценки рынка труда**: Анализ текущих трендов на рынке труда и выбор лучших направлений для обучения.

10. **Система мониторинга здоровья**: Экспертная система, которая анализирует данные пользователя (например, уровень активности, вес) и предлагает советы по улучшению здоровья.

11. **Определение уровня тревожности у студентов**: Система, которая на основе ответов на анкеты оценивает уровень стресса/тревожности и предлагает рекомендации.

12. **Онлайн-консультант по юридическим вопросам**: Экспертная система, предоставляющая базовые юридические советы по часто задаваемым вопросам.

13. **Система выбора оптимального маршрута**: Программа, которая принимает во внимание различные параметры и предлагает оптимальные маршруты до пункта назначения.

14. **Экспертная система для выбора электронной техники**: Рекомендации по выбору устройств (ноутбуки, смартфоны и т.д.) на основе предпочтений пользователя.

15. **Система оценки и улучшения учебной атмосферы**: Оценка факторов, влияющих на учебный процесс, и рекомендации по их улучшению.

16. **Консультант по экологии и устойчивому развитию**: Система, дающая советы по улучшению качества жизни, снижению углеродного следа и экосознанию.

17. **Тренажер по языкам с экспертной системой**: Программа, предлагающая задания на основе уровня знаний и предпочтений ученика.

18. **Анализ и прогнозирование исходов спортивных событий**: Система, использующая статистику и данные для прогнозирования результатов спортивных матчей.

19. **Система тестирования и рекомендаций при выборе книг**: Рекомендации книг на основе интересов, предыдущих чтений и оценок пользователя.

20. **Экспертная система для выбора программного обеспечения**: Консультант по выбору программного обеспечения на основе потребностей бизнеса или конкретного пользователя.

Изучение практических случаев применения ИИ в экспертных системах

Цели работы:

Развить практические навыки анализа и применения искусственного интеллекта в экспертных системах через исследование реальных кейсов, что позволит студентам понять их преимущества, недостатки и этические аспекты.

Задачи:

Ознакомиться с основами работы экспертных систем и их значением в различных областях. Проанализировать существующие экспертные системы, использующие искусственный интеллект, в таких сферах, как медицина, финансы и техническое обслуживание. Проанализировать влияние технологий ИИ на общество и рассмотреть этические вопросы, возникающие при разработке и внедрении экспертных систем.

Ожидаемые результаты

Студенты получат практический опыт применения теоретических знаний. Развить навыки критического мышления при анализе кейсов. Углубить понимание роли ИИ в экспертных системах и осознать важность этических аспектов.

Ход лабораторной работы

Подготовительный этап

1. **Введение в тему**:
    - Обсуждение целей и задач лабораторной работы.
    - Краткий обзор теоретических основ экспертных систем и их применения.
2. **Изучение материалов**:
    - Предоставление студентам списка экспертных систем с реальными примерами применения ИИ (например, медицинская

диагностика, финансовый анализ, техническое обслуживание и т.д.).
    - Рекомендации к оформлению результатов анализа.

Основной этап

3. **Групповой анализ случаев**:
    - Студенты делятся на группы (по 3-5 человек) и выбирают конкретный случай применения ИИ в экспертных системах.
    - Каждая группа анализирует ключевые аспекты выбранного проекта: цель, методы, технологии, результаты, сильные и слабые стороны. Для этого находят информацию в научных изданиях, интернете.
    - Готовят презентацию по выбранному кейсу для остальных групп, в своей работе разрешено использовать нейросети для написания текстов, генерации изображений, музыки, видео и презентации.
4. **Обсуждение и оценка**:
    - Каждая группа представляет результаты анализа своего случая и приводит ссылки на источники информации, которыми они пользовались.
    - Обсуждение особенностей и применения изученных экспертных систем.
    - Оценка, какие технологии и подходы показали наилучшие результаты и почему.
5. **Дискуссия по этическим аспектам**:
    - Обсуждение этических вопросов, связанных с использованием ИИ в экспертных системах.
    - Как технологии влияют на общество, что нужно учитывать при разработке и внедрении таких систем.

Заключительный этап

6. **Подведение итогов**:

- Обсуждение извлеченных уроков из практических случаев: что сработало, а что нет.
- Оценка значимости ИИ в современных экспертных системах.

7. **Формирование рекомендаций**:
   - Разработка рекомендаций по улучшению проектирования и внедрения ИИ в экспертные системы, основанных на проанализированных случаях.
8. **Обратная связь**:
   - Опрос студентов о том, что они извлекли из лабораторной работы.
   - Обсуждение предложений по улучшению формата или содержания лабораторной работы на будущее.

Примеры экспертных систем для анализа. Студенты сами могут предложить экспертную систему.

1. **IBM Watson для диагностики заболеваний**:
   - Кейс исследования применения IBM Watson в медицине для диагностики и подбора лечения на основе анализа больших объемов медицинского данных. Как система помогает врачам в сложных случаях?
2. **Системы Сбера для обнаружения мошенничества**:
   - Анализ системы Сбера, которая использует модели машинного обучения для обнаружения и предотвращения финансового мошенничества в реальном времени. Какие алгоритмы используются для выявления мошеннических схем?
3. **SAP Integrated Business Planning для управления запасами**:
   - Кейс о том, как SAP Integrated Business Planning помогает компаниям оптимизировать управление цепочками поставок и прогнозирование спроса с помощью ИИ. Как система снижает издержки и повышает эффективность?

4. **Recommender Systems от Amazon для персонализированных рекомендаций**:
    - Исследование работы рекомендательных систем Amazon, которые используют совместную фильтрацию и анализ поведения пользователей для предложений товаров. Как алгоритмы улучшают пользовательский опыт?
5. **Zendesk для автоматизации обслуживания клиентов**:
    - Кейс рассмотрения платформы Zendesk, использующей ИИ и чат-ботов для автоматизации обработки запросов клиентов. Как система улучшает коммуникацию и предоставляет пользователям быстрые ответы?
6. **Использование Google** ИИ для автоматизации тестирования своих программных решений.
    - Кейс применения алгоритмов машинного обучения для оптимизации генерации тестов, что позволяет минимизировать человеческие ошибки и увеличить скорость разработки.
7. Успешное применение **Microsoft** ИИ в своих системах для формальной верификации, например, в проекте **Azure**.
    - Кейс использования алгоритмов для проверки корректности программного обеспечения, что позволяет улучшить безопасность и надежность их облачных сервисов.
8. **Кейс** использования **Facebook** ИИ в инструментах для анализа кода, таких как **Phabricator**.
    - Это помогает разработчикам автоматизировать процессы ревью кода, выявляя потенциальные проблемы и улучшая качество кода еще на этапе его написания.
9. **Waymo**, подразделение **Google**, активно использует ИИ в тестировании и верификации программного обеспечения для автономных автомобилей.

- Кейс применения технологии ИИ для обработки огромных объемов данных, собранных во время тестирования, что помогает выявлять возможные баги и оптимизировать алгоритмы вождения.

Пример решения кейса.

Qualcomm использует методы машинного обучения для оптимизации проектирования интегральных схем. Они применяют ИИ для анализа больших данных, получаемых от предыдущих проектов, что помогает значительно ускорить процесс разработки новых чипов.

Ангализ кейса Qualcomm по применению методов машинного обучения для оптимизации проектирования интегральных схем.

1. Цель

- **Основная цель**: Ускорение и улучшение процесса проектирования интегральных схем (ИС) с высокой производительностью и меньшими затратами времени и ресурсов.
- **Дополнительные цели**: Увеличение точности проектирования, снижение количества ошибок и улучшение качества конечных продуктов.

2. Методы

- **Машинное обучение (МО)**: Применение алгоритмов обучения на основе больших данных для идентификации паттернов и оптимизации проектных решений.
- **Анализ данных**: Использование методов анализа данных для изучения результатов предыдущих проектов, что позволяет выявлять тенденции и предсказывать результаты.
- **Оптимизация**: Применение алгоритмов оптимизации для поиска наилучших конфигураций интегральных схем и улучшения их характеристик.

3. Технологии

- **Искусственный интеллект (ИИ)**: Включает в себя нейронные сети, деревья решений и другие алгоритмы МО, соответствующие задачам проектирования.
- **Большие данные**: Технологии для обработки и анализа больших объемов данных, таких как Apache Hadoop или Spark.
- **Системы проектирования**: Использование специализированного программного обеспечения для проектирования ИС, интегрированного с ИИ-инструментами.

4. Результаты

- **Ускорение разработки**: Снижение времени, необходимого для проектирования новых ИС, что позволяет быстрее выводить новые продукты на рынок.
- **Улучшение качества**: Уменьшение количества ошибок и дефектов в проектировании, что приводит к меньшему числу доработок.
- **Повышение эффективности**: Оптимизация использования ресурсов и снижение затрат на проектирование.

5. Сильные стороны

- **Инновации**: Применение передовых технологий, что позволяет Qualcomm оставаться на передовой в индустрии.
- **Конкурентоспособность**: Ускорение разработки и высокое качество продуктов позволяют компании удерживать конкурентные позиции на рынке чипов.
- **Клиентоориентированность**: Удовлетворение растущих потребностей клиентов в более производительных и энергоэффективных чипах.

6. Слабые стороны

- **Сложность внедрения**: Интеграция ИИ и МО в существующие процессы может столкнуться с сопротивлением сотрудников или требовать значительных инвестиций в обучение и изменение культуры.
- **Зависимость от данных**: Эффективность алгоритмов машинного обучения напрямую зависит от количества и качества имеющихся

данных, что может стать препятствием, если данные неполные или искаженные.

- **Риск переобучения**: Алгоритмы могут переобучаться на специфических данных, что приведет к снижению их эффективности на новых проектах.

Таким образом, проект Qualcomm демонстрирует активное использование ИИ и МО для оптимизации разработки интегральных схем, что приносит множество преимуществ, но также сопряжен с определенными рисками и вызовами.

Решение слабых сторон проекта Qualcomm можно рассмотреть через несколько стратегий и подходов:

1. Сложность внедрения

- **Обучение и подготовка персонала**: Проведение регулярных тренингов и семинаров для сотрудников, чтобы повысить их квалификацию в области ИИ и МО. Это также поможет снизить сопротивление изменениям.
- **Создание междисциплинарных команд**: Формирование команд, состоящих из специалистов в области проектирования и данных, чтобы облегчить интеграцию новых методов и технологий в существующие процессы.
- **Поэтапное внедрение**: Начинать с малых пилотных проектов и поэтапно расширять масштабы внедрения, что позволит лучше адаптировать процессы и минимизировать риски.

2. Зависимость от данных

- **Расширение источников данных**: Активный поиск и использование новых источников данных, как внутренних, так и внешних, для повышения качества и объема доступной информации.

- **Предобработка данных**: Внедрение процессов очистки и предобработки данных, чтобы улучшить их качество перед использованием для обучения моделей.
- **Создание систем управления данными**: Разработка эффективных систем хранения и управления данными для обеспечения их доступности и качества.

3. Риск переобучения

- **Кросс-валидация**: Использование методов кросс-валидации для проверки устойчивости моделей на разных наборах данных, что поможет предотвратить переобучение.
- **Регуляризация**: Применение техники регуляризации, чтобы снизить риски переобучения моделей, сохраняя их способность делать обоснованные предсказания.
- **Мониторинг и обновление моделей**: Постоянный мониторинг производительности моделей и их регулярное обновление на основе новых данных или изменений в проектах.

4. Адаптация максимальных практик

- **Применение лучших практик из отрасли**: Изучение и внедрение успешного опыта компаний, использующих ИИ и МО в аналогичных областях, что может помочь избежать распространенных ошибок.
- **Совместимость стандартов**: Разработка и внедрение стандартов для работы с ИИ и МО, чтобы обеспечить согласованность и совместимость в проекте.

5. Тестирование и оптимизация

- **А/Б тестирование**: Проведение А/Б тестов для проверки различных подходов и выбора наилучших решений на основе статистических данных.
- **Обратная связь от пользователей**: Сбор обратной связи от команды проектировщиков и анализа реального использования моделей, чтобы вносить улучшения.

В работе были использованы материалы:

1. **Официальный сайт Qualcomm**: https://www.qualcomm.com (разделы "News" и "Technology").
2. **Блог Qualcomm**: https://www.qualcomm.com/blog (публикации о новых разработках и исследованиях).
3. **Страница с исследованиями на ResearchGate**: https://www.researchgate.net (поиск по "Qualcomm AI").
4. **YouTube-канал Qualcomm**: https://www.youtube.com/user/Qualcomm (видеоматериалы о новых технологиях и применениях ИИ).
5. **Статьи на TechCrunch**: https://techcrunch.com (статьи о Qualcomm и ИИ).