Министерство просвещения Российской Федерации
ИНСТИТУТ СОДЕРЖАНИЯ
И МЕТОДОВ ОБУЧЕНИЯ
имени В.С. ЛЕДНЕВА

Методические рекомендации по изучению основ искусственного интеллекта в учебном предмете «Информатика» основного общего образования и внеурочной деятельности

МОСКВА
2025

УДК 37.02
ББК 74.202

Авторы:
Н.Н. Самылкина, доктор педагогических наук, доцент, ведущий научный сотрудник Центра математического и естественно-научного общего образования ФГБНУ «Институт содержания и методов обучения имени В.С. Леднева»

А.А. Салахова, кандидат педагогических наук, доцент кафедры теории и методики обучения математики и информатики ИМИ МПГУ

Рецензенты:
И.В. Левченко, доктор педагогических наук, профессор, профессор кафедры теории и методики обучения математики и информатики ИМИ МПГУ

Н.И. Волынчук, кандидат педагогических наук, заведующий Центром математического и естественно-научного общего образования ФГБНУ «Институт содержания и методов обучения им. В.С. Леднева»

Корректор:
Е.А. Семенихина

Методические рекомендации по изучению основ искусственного интеллекта в учебном предмете «Информатика» основного общего образования и внеурочной деятельности / Н.Н. Самылкина, А.А. Салахова, Москва: ФГБНУ «Институт содержания и методов обучения им. В.С. Леднева», 2025. – 78 с.

ISBN 978-5-6055604-6-3

Методические рекомендации по изучению основ искусственного интеллекта в учебном предмете «Информатика» основного общего образования и внеурочной деятельности будут полезны учителям информатики. В них описаны общие подходы к интеграции тематики искусственного интеллекта (ИИ) в основное общее образование, примерное содержание тематики ИИ по тематическим разделам учебного предмета «Информатика», а также содержание модуля ИИ в тематических разделах «Цифровая грамотность», «Теоретические основы информатики» и «Алгоритмы и программирование».
Методические рекомендации разработаны в рамках государственного задания ФГБНУ «Институт содержания и методов обучения им. В.С. Леднева» на 2025 год «Обновление содержания общего образования».

ISBN 978-5-6055604-6-3

УДК 37.02
ББК 74.202

Оглавление

Введение.....3
Общие подходы к интеграции тем ИИ в основное общее образование 4
Примерное содержание тематики искусственного интеллекта по тематическим разделам учебного предмета «Информатика».....6
Содержание модуля ИИ в тематическом разделе «Цифровая грамотность» 9
Содержание модуля ИИ в тематических разделах «Теоретические основы информатики» и «Алгоритмы и программирование».....20
Содержание модуля ИИ в тематическом разделе «Информационные технологии» 32
Приложение 1. Вклад отечественных ученых в развитие искусственного интеллекта.....41
Приложение 2. Организация курса по выбору или внеурочной деятельности для подготовки к олимпиадам по ИИ 45

Методические рекомендации по изучению основ искусственного интеллекта в учебном предмете «Информатика» основного общего образования и внеурочной деятельности

Введение

Активное использование продуктов и технологий искусственного интеллекта (ИИ) в профессиональной деятельности, системе образования, принятие государственных программ, связанных с развитием ИИ и анализа больших данных актуализируют проблему подготовки кадров в национальной системе образования. Фундаментом подготовки необходимых кадров для цифровой экономики России является общеобразовательная подготовка школьников. На тематику изучения основ искусственного интеллекта ориентируется школьный курс информатики, начиная с использования технологий искусственного интеллекта в основном общем образовании до теоретических и практических основ в содержании информатики углубленного уровня среднего общего образования.

Вместе с тем, использование искусственного интеллекта в составе инструментальных средств обучения по различным учебным предметам и внеурочной деятельности необходимо осуществлять продуманно, на основе общепринятых этических норм и проработанного нормативного регулирования его использования в образовательной практике. В содержании информатики необходимо акцентировать внимание на фундаментальную составляющую тематики искусственного интеллекта: методы моделирования, интеллектуальные методы и их реализацию в алгоритмах, программирование на профессиональных языках и пр. Роль учителя – ведущая, а инструменты ИИ – средство развития обучающихся и инструмент для организации взаимодействия в информационной образовательной среде.

Методические рекомендации по изучению основ искусственного интеллекта в учебном предмете «Информатика» основного общего

образования и внеурочной деятельности позволят учителю информатики и администрации организации, реализующей программы основного общего образования, рассмотреть возможность интеграции тем искусственного интеллекта и тематических разделов учебного предмета «Информатика», а также внеурочной и проектной деятельности учащихся основного общего образования.

Общие подходы к интеграции тем ИИ в основное общее образование

В основной образовательной программе каждой школы учебный предмет «Информатика» обязателен в 7–9 классах и его можно изучать на базовом[1] или углубленном уровнях[2]. По структуре и последовательности изучения курса он синхронизирован на обоих уровнях. Это означает, что второй час информатики из части учебного плана, формируемой участниками образовательных отношений, на углубленном уровне позволяет отработать на большем количестве заданий и практических работ изучаемую тему. Учащиеся именно так выходят на функциональный уровень усвоения материала, в отличие от уровня понимания и представлений, который обеспечивает базовый уровень. Фактически содержательная составляющая практически одинаковая на двух уровнях изучения материала, а деятельностная – усиливается. Содержательная часть тематики ИИ только незначительно расширяет фундаментальный материал курса информатики в части обсуждения дополнительных понятий и связанных с ними видов

[1] Рабочая программа учебного предмета «Информатика», базовый уровень (для 7–9 классов образовательных организаций); одобрена решением федерального учебно-методического объединения по общему образованию, протокол № 3/21 от 27.09.2021 г. URL: https://fgosreestr.ru/uploads/files/dcca994c21165f0d49d4baf4a7e008c0.pdf?ysclid=lrngryssgv793769966 (дата обращения: 12.01.2025).

[2] Рабочая программа учебного предмета «Информатика», углубленный уровень (для 7–9 классов образовательных организаций); одобрена решением федерального учебно-методического объединения по общему образованию, протокол № 2/22 от 29.04.2022 г. URL: https://shkola5pytyax-r86.gosweb.gosuslugi.ru/netcat_files/30/69/INFORMATIKA_uglublennyy_uroven_Realizatsiya_trebovaniy_FGOS_OOO.pdf (дата обращения: 12.01.2025).

деятельности, относящихся к ИИ, а также использования в жизни и образовании новых технологических инструментов, реализованных в виде специализированного ПО. Углубленный уровень от базового уровня изучения информатики основного общего образования в части освоения тематики ИИ отличается возможностью решения большего количества задач, а также частью учебного плана, формируемой участниками образовательных отношений, в котором возможны: реализация проектной деятельности по тематике ИИ, систематическая подготовка к олимпиадам по ИИ, робототехническая подготовка для последующего перехода к интеллектуальной робототехнике.

Программа учебного предмета «Информатика» должна обязательно содержать тематику искусственного интеллекта и больших данных как обязательного компонента современных информационных технологий. У учащихся должна быть также возможность изучать интересующую тематику во внеурочной и проектной деятельности. Логика предложения и развертывания тем искусственного интеллекта и больших данных должна учитывать возможности их усвоения обучающимися разного возраста и способностей, т.е. представлена на двух уровнях усвоения в контексте учебного предмета, а также может быть реализована в виде самостоятельного тематического блока учебного предмета «Информатика» для возможной реализации курса по выбору участников образовательных отношений. Данный курс может обеспечивать, например, подготовку к тематической олимпиаде.

При планировании курса информатики следует принять во внимание, что одночасовой базовый уровень изучения информатики в основном общем образовании предельно насыщен и не позволит рассматривать отдельно темы ИИ [1]. На базовом уровне можно только затронуть на примерах тематику ИИ и больших данных. Например, развивая обсуждение новых понятий в сравнении (чем отличаются понятия «данные» и «большие данные»; «интеллект» и «искусственный интеллект», «поиск информации» и

«интеллектуальный поиск»?), либо – используя на практике интеллектуальные приложения для генерации текста, изображений, голоса с пояснением, за счет чего эти возможности реализованы в программах. Для создания контента, написания кода или анализа текстов используются специальные программы — YandexGPT или GigaChat. Если же требуется создание изображений, применяются — Кандинский, Шедеврум.

Индивидуальный интерес к теме ИИ для изучающих информатику на базовом уровне возможно поддержать только в проектной деятельности учащихся и привлечением к дополнительным активностям в виде экскурсий, интеллектуальных игр (квестов, квизов и пр.). В основной школе пока еще есть время у учащихся сориентироваться в выборе будущего профессионального трека, нарастить изучение программирования при необходимости.

Самым серьезным дополнением к изучению тематики искусственного интеллекта в курсе информатики основного общего образования углубленного уровня является расширенное изучение языка программирования Python для подготовки к олимпиадам по ИИ как одного из профилей Всероссийской олимпиады школьников по информатике, а также изучение робототехники, также являющегося с 2025 года одним из профилей ВсОШ. Большинство образовательных организаций выделяют на это отдельное время за счет части учебного плана, формируемой участниками образовательных отношений.

Примерное содержание тематики искусственного интеллекта по тематическим разделам учебного предмета «Информатика»

На базовом уровне изучения информатики основного общего образования акценты направлены на использование цифровых инструментов в образовательных целях, которые оцениваются в комплексе с формированием фундаментальных предметных результатов курса информатики и метапредметных результатов.

На углубленном уровне изучения информатики по тематическим разделам к изучаемому материалу добавляются новые понятия и задания, относящиеся к тематике ИИ.

Примерное содержание тематики ИИ для углубленного изучения, интегрированное с темами курса информатики основного общего образования, представлено в таблице 1.

Таблица 1

Примерное содержание тематики ИИ для углубленного изучения, интегрированное с темами курса информатики основного общего образования

Раздел	**Углубленный уровень изучения информатики**	**Внеурочная деятельность и проекты**
Цифровая грамотность	Разделение понятий «информация», «знания», «данные», «большие данные» в контексте изучения возможностей современных компьютеров. Современные сервисы интернет-коммуникаций. Правовые аспекты использования программного обеспечения с интеллектуальными сервисами	Межпредметные проекты с использованием анализа и визуализации данных в электронных таблицах. Виртуальная экскурсия в музей старых компьютеров и до суперкомпьютеров; создание плакатов со шкалой времени для изучения истории развития компьютеров и появления ПО с интеллектуальной составляющей.
Теоретические основы информатики	Обзор интеллектуальных сервисов для обработки текста (программы-переводчики), звука (чат-боты), графики и видео (распознавание образов, задачи классификации и кластеризации). Графы (выход на дерево решений). Основы моделирования (расширяются примеры моделей за счет математической модели нейронной сети)	Использование интеллектуальных сервисов в междисциплинарных проектах

Раздел	Углубленный уровень изучения информатики	Внеурочная деятельность и проекты
Информационные технологии	Работа с указанными сервисами (создание промптов, перевод, распознавание). Использование чат-ботов. Приемы анализа и визуализации данных	Создание школьного цифрового ресурса: создание образа виртуального блогера, ведение блога с текстами и иллюстрациями, созданными генеративными нейронными сетями, озвучка и создание сгенерированного видео по теме
Алгоритмы и программирование	Изучение возможностей языка Python (обработка строк, символов стандартными функциями). Интеллектуальная робототехника	Подготовка к олимпиадам по искусственному интеллекту. Интеллектуальная робототехника

В результате анализа федеральных рабочих программ по информатике основного общего образования базового и углубленного уровня было отмечено, что тематика искусственного интеллекта обязательно входит в курс информатики как перспективное направление информационных технологий. В основном общем образовании делаются акценты на применение различных программ и приложений с интеллектуальными возможностями для решения образовательных и повседневных задач. В ходе использования таких программ происходит знакомство с историческими сведениями о развитии технологий ИИ, с терминологией и потенциальными возможностями современного программного обеспечения и различных сервисов. Необходимый содержательный контент встраивается в темы курса информатики там, где это возможно и необходимо сделать. В условиях ограниченного времени на изучение информатики базового уровня (1 ч/н) – это затруднительно реализовать в полном объеме, все будет зависеть от возможностей учащихся и выбора учителя информатики. Поэтому предлагается содержание (теория и задания) для углубленного изучения

информатики (2 ч/н), из которого учитель может отбирать материал для обзорного изучения темы ИИ на базовом уровне. Содержание разбито по тематическим разделам и внутри раздела предложена тема, которая наилучшим образом подходит для расширения в части основ искусственного интеллекта и анализа данных. Часто тематика ИИ объединяет все разделы, практически в любом из них можно делать обобщения и отсылки по теме.

Содержание модуля ИИ в тематическом разделе «Цифровая грамотность»

Темы раздела «Цифровая грамотность» изучаются в 7 и 9 классах в соответствии с тематическим планированием. При изучении темы «История развития компьютеров и программного обеспечения» можно добавить сведения об истории появления понятия «искусственный интеллект», а также «нейронная сеть» и «интеллектуальные сервисы». При объяснении развития компьютерной техники нужно акцентировать внимание на отдельные факты, например, следующие:

- термин «искусственный интеллект» в 1956 году ввел Джон Маккарти, известный как создатель языка Lisp;
- термин «нейронная сеть» не является новым: он также появился в 1943 году в виде идеи, предложенной Уорреном Мак-Калоком и Уолтером Питтсом, а реализована нейронная сеть впервые на практике была Фрэнком Розенблаттом в конце 1950-х;
- искусственный интеллект раньше использовался для нужд лабораторий, различных специализированных вычислений и куда реже для коммерческих целей, поэтому о нём практически не говорили;
- советские исследователи также занимались вопросами искусственного интеллекта; например, в 1960-х велись отечественные разработки программ компьютерного зрения по распознаванию простых геометрических объектов – начало становления компьютерного зрения наряду с попытками распознавать рукописный текст.

Для расширения темы и более подробного рассмотрения вклада российских ученых в разработку ИИ используйте Приложение 1.

Необходимо встроить появление искусственного интеллекта и его развитие как часть истории развития компьютеров и программного обеспечения при изучении этой темы согласно планированию.

При обсуждении новых понятий не обязательно ориентироваться на их научные определения, для основного общего образования достаточно понимания смысла.

Искусственный интеллект – наука и технология, которая позволяет создавать интеллектуальные машины и программы. Эти программы и системы, в которые они входят, способны заменять человека, где раньше это было невозможно. Чаще всего они умеют анализировать большое количество информации (например, выявлять неочевидные при анализе простыми статистическими алгоритмами закономерности), предоставляя выводы и предлагая решения. Важно понимать, что конечное решение всегда принимает только человек.

Традиционно в изучении искусственного интеллекта было два больших направления – в одном упор делался на умения рассуждать согласно логическим правилам, а во втором – на попытку воспроизвести строение и работу нейронной системы и мозга человека.

Нейронная сеть – это компьютерная программа, имитирующая биологические принципы работы человеческого мозга. Она состоит из множества нейронов, объединенных в группы (слои). Нейрон – это особая клетка в организме, которая обрабатывает получаемый от других нейронов и систем электрический сигнал. Каждый нейрон обрабатывает сигнал по-своему. Для компьютерного нейрона это означает, что каждый такой элемент получает на вход некие данные, применяет к ним специфическую для этого нейрона функцию (формулу для преобразования) и передает обработанный сигнал дальше (рис. 1).

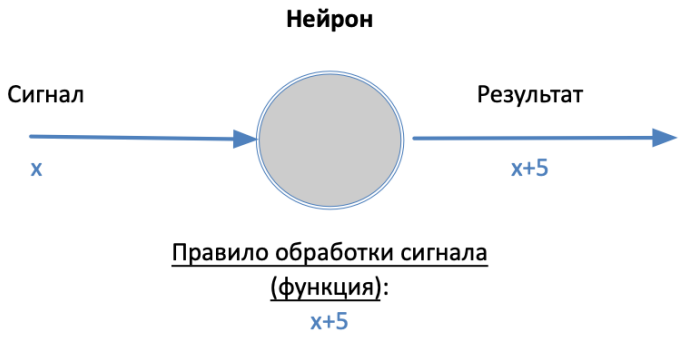

Рис. 1. Модель нейрона

Нейроны объединяют в слои в зависимости от источника сигнала и цели применения функции на них (рис. 2).

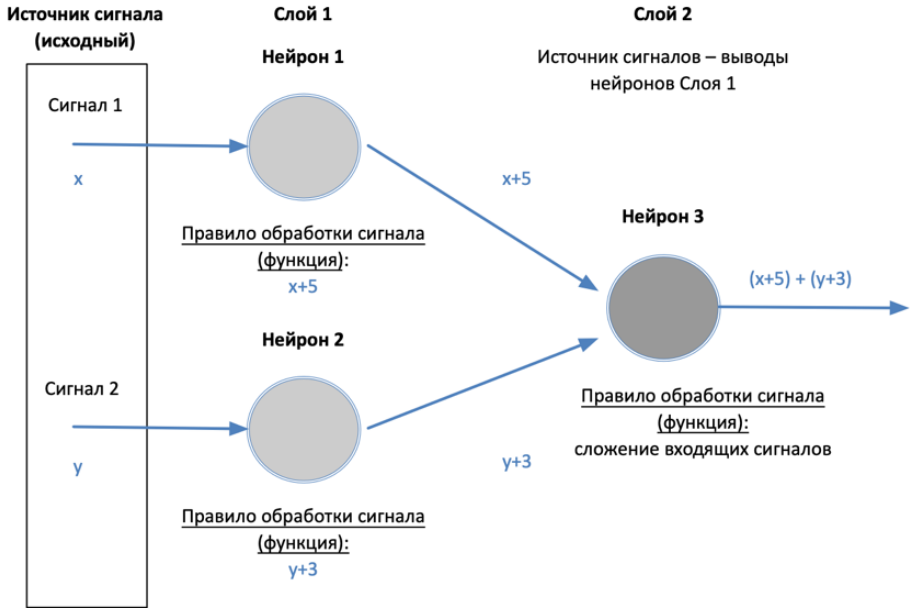

Рис. 2. Модель нейронной сети

Модель нейронной сети можно рассматривать в теме «Моделирование» тематического раздела «Теоретические основы информатики».

Важно уточнить, что компьютеры не обладают сознанием. Даже если мы моделируем программу, действующую по правилам логического мышления, мы не можем сделать так, чтобы программа понимала смысл исходя из контекста и набора данных.

Задание 1. Придумайте вопросы по схеме Терри Винограда для голосового помощника. Например, «Дочка убедила купить компьютер свою маму, когда та вырастет. Кто вырастет?». Проанализируйте ответ вместе с учащимися. В предложении могут быть допущены ошибки специально. Суть

теста заключается в том, что для человека логически понятно происходящее, и он не должен опираться на конкретные местоимения или формулировку.

Сегодня мы часто используем **интеллектуальные сервисы** – это различные веб-приложения и иные программы со встроенными интеллектуальными алгоритмами, настроенными для эффективного решения задач в ограниченной области (например, интеллектуальные сервисы генерации картинок). К сожалению, область работы таких приложений всегда ограничена.

Также интеллектуальные сервисы встраиваются как компоненты и надстройки в привычные нам программные продукты. Например, в виде плагинов для браузеров, используемых для работы с текстом, улучшения ранжирования поисковых результатов и многого другого. Подробнее об интеллектуальных сервисах мы поговорим в разделе «Информационные технологии». В 7 классе рассматриваются такие темы, как «Поисковые системы. Поиск информации по ключевым словам и по изображению», «Стратегии безопасного поведения в Интернете», «Современные сервисы интернет-коммуникаций», а на 9 класс остается тема «Правовая охрана программ и данных». При знакомстве с поисковыми сервисами уточнить, что сегодня мы чаще всего используем интеллектуальные поисковые системы, которые:

- умеют обрабатывать естественный язык (без строгих правил задания запросов, как это было ранее в поисковых системах старого поколения);
- обрабатывают запрос по близкому смыслу через поиск похожих понятий внутри заданной области знаний (а не прямое соответствие запросу – ключевым словам);
- ранжируют результаты в зависимости от поведения самого человека в Сети на разных сайтах и в приложениях, делая персональные подборки.

Современные интеллектуальные поисковые системы используют интеллектуальные сервисы, которые также умеют обобщать результаты поисковой выдачи в единый ответ на естественном языке с указанием

источников. Однако важно понимать, что часто источники могут указываться недостоверно, поскольку программа не способна оценить их смысл. Они требуют обязательной проверки человеком.

Также интеллектуальные сервисы позволяют осуществлять поиск с помощью функции распознавания образов на картинке или с камеры телефона.

При объяснении темы в качестве демонстрационного материала необходимо применение отечественного ПО, включая реализации нейронных генеративных (умеющих создавать новые данные с заданными характеристиками) сетей с серверами, расположенными в РФ.

Здесь важно понимать смысл перечисленных терминов в ходе выполнения практических заданий.

Задание 2. С помощью интеллектуальной поисковой системы найти информацию и подготовить небольшую заметку (до 2 страниц) о советских ЭВМ. Можно изменить тему по усмотрению учителя. Обязательно указание источников. Для выполнения данного задания можно воспользоваться режимом «Нейро» (рис. 3) в поисковой системе «Яндекс».

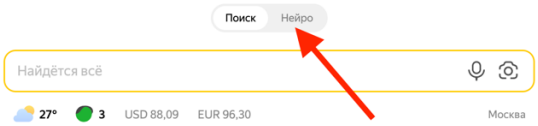

Рис. 3. Режим «Нейро» в поисковой системе «Яндекс»

Задание 3. С помощью встроенных поисковых помощников на мобильных устройствах обучающихся или приложения «Яндекс» выполнить поиск с камеры, наведя ее на учебник по информатике. Какие результаты будут предложены? Попробовать произвести поиск с разных ракурсов и при разной освещенности. Что влияет на результат? Какие выводы можно сделать?

В качестве иллюстративного примера можно привести поиск по картинке на маркетплейсах.

Далее проводится аналогия поиска по картинкам и поиска по лицам. В обоих случаях сравнивается записываемая в формате графического файла информация.

Распознавание лиц – это один из примеров интеллектуального поиска по картинке. Поиск и распознавание лиц осуществляется следующим образом:

1. Производится получение биометрического образца (размеры и расстояния между элементами лица). Здесь определяется положение лица на фотографии или в видео. Интеллектуальные системы могут произвести анализ на видео и изображениях там, где человек не смотрит прямо в камеру.

2. Если качество картинки достаточное, например, лицо освещено и больше, чем наполовину, находится в кадре, то образец отправляется на обработку. Он масштабируется и очищается от цвета, который может зависеть, например, от освещения.

3. Полученный образец сравнивают с базой данных, имеющейся в системе. Такие базы данных с лицами постоянно обновляют.

Распознавание на видео происходит похожим образом: распознаются лица на отдельных кадрах. Так работают, например, камеры городских систем безопасности.

При рассмотрении темы «Стратегии безопасного поведения в Интернете» добавляем такие популярные термины как «**чат-бот**» и «**фейк**» (Deep Voice, Deep Fake). Возможно, что учащиеся самостоятельно озвучат значения этих терминов, от учителя потребуется только незначительное их уточнение.

В Интернет появился голосовой помощник – чат-бот (не всегда интеллектуальный, иногда он работает по строгому набору вопросов). Желательно обсудить правила общения с чат-ботами: не грубить, задавать только вопросы по теме. Как отличить чат-бот и человека?

В данном случае важно сказать, что интеллектуальные системы могут использоваться и злоумышленниками для имитации голоса или даже видео человека для совершения неправомерных действий.

Учителю важно дать обучающимся памятку действий:

- проверить входящий канал связи (телефон, смс, почта);
- в случае получения непонятной записи перезвонить человеку или воспользоваться другим способом связи с ним;
- обязательно предупредить взрослых.

В качестве демонстрационного материала учитель может показать небольшой отрывок сгенерированного собственного голоса с помощью свободно распространяемой модели XTTS или показать видеофрагмент работы российской нейросети VeraVoice, в котором создатели нейросети говорят о возможных опасностях применения таких технологий.

Не стоит приводить конкретные названия для моделей нейронных сетей, используемых злоумышленниками или пранкерами, чтобы не спровоцировать обучающихся к деструктивному опыту.

Еще один термин «**цифровой след**» стал популярным и его обсуждение внесет свой вклад в копилку стратегий безопасного поведения в сети. «***Цифровой след***» – это любые данные об активности пользователя в Сети, которые можно отследить**.** Чем больше наш цифровой след в интернете, тем больше информации мы предоставляем для создания ситуаций, где можно не только предугадать наше поведение (что чаще всего используется в рекламных целях), но и скопировать наши данные, привычки и многое другое. Например, для обучения модели клонирования голоса достаточно записи от 30 секунд, взятой из открытых бесед в социальных сетях или иных источников.

Сегодня цифровое поведение предусматривает еще большее внимание к собственным действиям, поскольку вся информация о нашей деятельности в сети Интернет анализируется для составления нашего профиля. Для целей рекламы он хранится в обезличенном виде, но его достаточно легко

идентифицировать с конкретным человеком. Такой след также позволяет рекомендательным интеллектуальным системам предлагать нам интересные лично нам фильмы, сериалы, музыку, тематические кафе, интересные покупки и многое другое. Однако большая часть цифрового следа пользователя не относится к его персональным данным и не защищается по закону. Рекомендуется обеспечить безопасность своего цифрового следа выбором надежных паролей, контролем за конфиденциальной информацией и осознанным использованием интернет-сервисов.

Темы «*Стратегии безопасного поведения в Интернете*» и «*Современные сервисы интернет-коммуникаций*» взаимосвязаны, поэтому их можно рассматривать вместе или менять последовательность. Здесь следует акцентировать внимание на достоверности полученной информации. Стоит отметить, что доступность информации, включая быстрый поиск и предоставление обобщенных результатов сразу интеллектуальной системой, а также применение генеративных алгоритмов[3], приводит к возможному искажению информации. Сегодня мессенджеры становятся одним из основных каналов информационного обмена, поэтому важно проверять достоверность полученных сведений через первоисточники.

Важно понимать, что такая информация не обязательно представлена в текстовом виде. Она может быть графической, видео или аудио.

Задание 4. В качестве задания можно привести примеры развлекательных или новостных каналов, предложив их вынести в таблицу на две колонки (достоверные или недостоверные источники).

Какие критерии использовали обучающиеся при выполнении задания? Можно ли доверять любому каналу? Указываются ли в таких каналах ссылки на источники?

Используйте только ПО, которое не требует регистрации или позволяет зарегистрироваться без передачи персональных данных.

[3] алгоритмы машинного обучения, основанные на методах и моделях, позволяющих создавать новые данные из имеющихся с аналогичными характеристиками.

Обратите внимание, что часть популярных сервисов доступна только с регистрацией аккаунта совершеннолетнего лица.

Также можно дополнить, что современные сервисы интернет-коммуникации стремятся стать не социальными сетями или мессенджерами, а ***суперприложениями*** – цифровыми платформами, объединяющими предоставление цифровых (часто и оффлайн) услуг разного характера, предлагаемых на основе поведения пользователя с помощью рекомендательных алгоритмов. Часто в суперприложения встроены голосовые помощники. Например, в VK встроена «Маруся». Иногда интеллектуальные рекомендательные чат-боты не имеют озвучки, как чат-бот – помощник портала «Госуслуги». Либо ассистент может быть и вовсе не персонифицирован.

Этическая сторона вопроса работы с данными, обрабатываемыми и предоставляемыми генеративными алгоритмами в различных приложениях, как основа будущих правовых решений в информационной сфере, может быть обсуждена в 9 классе в рамках темы «Правовая охрана программ и данных» или в контексте любой темы сетевой безопасности.

Существует несколько аспектов этического использования генеративных алгоритмов и данных, которые они обрабатывают или предоставляют в качестве предлагаемого решения (результата их работы).

Первый вопрос довольно очевиден – на каких данных обучена нейросеть? Было ли получено разрешение на использование этих данных? Нейросети не умеют создавать новое. Именно поэтому нельзя сказать, что нейросеть занимается творчеством. Любой результат работы генеративного алгоритма – это комбинация маленьких отрывков или неделимых исходных элементов (***батчей***), полученных из обучающего материала. Результат работы генеративного алгоритма не является полностью оригинальным произведением. Зачастую владельцы могли не давать согласия на использование своих произведений, текстовых или графических, для обучения. Например, именно поэтому случился большой скандал и

забастовка цифровых художников в 2023 году. Тем более нельзя сказать, что автором полученного результата является тот, кто запросил генерацию этого контента.

Но иногда алгоритмы обучаются на данных, которые им предоставил сам пользователь. Эти данные разделяются на батчи, которые впоследствии могут использоваться в системе, либо могут быть применены выявленные закономерности. К сожалению, иногда этими обрабатываемыми данными могут становиться даже секретные материалы! Они могут составлять коммерческую или иную тайну. Кроме того, если мы используем неизвестных поставщиков услуг, то мы не можем точно знать, что будет происходить с нашими данными.

Второй вопрос не менее важен. Кто несет ответственность за результаты работы нейросети? Кто принимает решение о публикации и приведении в действие решений, предложенных искусственным интеллектом? Нейросеть анализирует закономерности, возникающие, например, в текстах на определенном языке. Она не может оценивать влияние результата на моральное состояние читателя, предугадывать последствия публикации недостоверной информации. Например, какие последствия может иметь пост о победителе важного конкурса со сгенерированным и никогда не существовавшим персонажем? А если в предложенном рецепте салата были несъедобные сочетания трав и специй?

Кроме того, мы можем столкнуться с примером феномена *галлюцинации нейросети*. Именно он описан выше. В этом случае генеративные алгоритмы предлагают описания не существовавших исторических событий, персон, авторов и произведений на основе часто встречаемых и хорошо сочетающихся (по анализу зависимостей) батчей. То есть нейросеть предлагает то, что могло бы быть достаточно достоверным по формальным признакам.

Конечное решение о применении предложенного варианта, как и ответственность за это решение, целиком и полностью лежит на человеке.

И еще один дополнительный вопрос, который также можно рассмотреть: кому принадлежит итоговый ответ нейросети. Например, права на картинки, сгенерированные в сервисе «Шедеврум», принадлежат правообладателю платформы — компании «Яндекс». Это означает, что пользователь может использовать их исключительно в личных некоммерческих целях.

Задание 5. Составить таблицу с перечислением трех интеллектуальных сервисов с указанием, кому принадлежат права на результат их работы.

Также важно уточнять, какие риски существуют при ***облачном хранении данных***. Кому они принадлежат? Есть ли доступ у провайдера услуг к этим данным?

Важно отметить, что для предоставления своих данных в Интернет не обязательно размещать свои документы на облаке в явном виде. Сегодня в ПО часто встраивают функции, которые производят вычисления на облаке, предоставляя результат пользователю. Следует понимать, что нет принципиальных отличий в отношении загрузки и обработки пользовательских данных между веб-сервисом, представленным в виде сайта или в виде дополнительной функции внутри классического приложения на компьютере.

Примеры дополнительных заданий (для домашнего задания и последующего обсуждения в классе, для текущего оценивания усвоения теоретического материала и т.д.)

1. Составить шкалу времени появления понятий «искусственного интеллекта», «нейронной сети», «цифрового следа», «чат-ботов» и самых известных программ и сервисов.
2. Подготовить с помощью интеллектуальной поисковой системы справку о первых нейронных сетях.
3. Найти с помощью интеллектуального поиска камерой информацию об объекте.
4. Составить подборку достоверных новостей по теме «Компьютерные технологии» с указанием источников.

5. Задача-ситуация. Вам позвонила бабушка, голос было плохо слышно. Она сказала, что ей срочно нужны деньги, она пришлет смс, но родителям говорить не обязательно, она сама со всем справится. Вы за неё очень переживаете. Описать порядок своих действий.

6. По предоставленной записи диалога определить, был он с человеком или с чат-ботом. Выписать критерии, по которым было определено. Имеет ли значение, кто отвечал?

7. С помощью чат-бота Алиса от «Яндекс» выполнить поиск информации о чат-боте Eliza.

8. Подготовить инфографику об элементах цифрового следа (схема).

9. Практическая работа. Выполнить поиск по запросу «школа». Сравнить результаты, полученные в одном и том же поисковике в обычном режиме, с включенной геопозицией на телефоне или ином гаджете, в режиме «Инкогнито». Выписать различия и факторы, которые могли повлиять на их появление.

10. Посмотреть ролик, созданный с помощью технологии Deep Fake. Сформулировать и выписать в формате памятки критерии, на которые стоит обращать внимание при распознавании сгенерированного голоса или видео.

11. Подготовить сообщение о применении элементов искусственного интеллекта в используемых приложениях, включая операционные системы, текстовые редакторы и браузеры.

12. Подготовить сообщение об использовании рекомендательных сервисов для персонализации контента: от образования до развлечений.

Содержание модуля ИИ в тематических разделах «Теоретические основы информатики» и «Алгоритмы и программирование»

К разделу «Теоретические основы информатики» можно отнести обсуждение новых понятий, относящихся к искусственному интеллекту и каким-то образом связанных с этой темой, а также модель нейронной сети из

предыдущего раздела, если вы её не использовали ранее. Новые понятия только немного дополняют изучаемые по программе темы, поэтому последовательность введения новых понятий отслеживается учителем. Понятия вводятся в контексте объяснения основной темы, являются скорее актуальным фоном и не подлежат обязательному контролю в основной школе.

Раздел «Алгоритмы и программирование» дополнен темой искусственного интеллекта меньше всего, поскольку в основной школе предусматривается только знакомство с основными алгоритмическими конструкциями и алгоритмами на их основе, а также основами одного текстового языка программирования.

Стоит отметить, что знакомство с искусственным интеллектом на языках учебных исполнителей (КУМИР) и подобных в принципе невозможно ввиду того, что данный язык скорее является набором команд для ограниченного перечня исполнителей (причем для каждого из исполнителей предусмотрены собственные команды). Погружение в тему искусственного интеллекта в основной школе с помощью языка Python как инструмента также затруднительно, поскольку для работы с уже готовыми решениями требуется уметь выбирать, подключать и использовать сторонние библиотеки, а также иметь более глубокое понимание поиска, сортировки и особенностей представления данных в других форматах, также желательно понимать, как можно работать с таблицами.

Но в том случае, когда программирование изучалось ранее, и учащимся предусматривается прагматическая цель – участие в профиле «Искусственный интеллект» Всероссийской олимпиады школьников по информатике, надо к ней целенаправленно готовиться. Для организации такой подготовки необходим курс по выбору участников образовательных отношений в вариативной части учебного плана или курс внеурочной деятельности. Содержание курса могут составлять задания из Приложения 2.

Большая часть представленных здесь заданий может быть использована в разделе «Теоретические основы информатики» с выходом на раздел «Алгоритмы и программирование». Например, понимание «потока данных» при работе с большими данными продолжит обсуждение понятий «информация», «данные» и «большие данные». Изучение двоичного поиска в упорядоченном массиве позволит лучше понять использование графов и выйти на актуальную тему, связанную с ИИ, – бинарные деревья принятия решений, которую можно связать с классической темой в программировании – рекурсией.

При обсуждении понятий «**информация**», «**данные**» и «**большие данные**» можно расширить круг обсуждаемых вопросов за счет характеристик больших данных и обработки потока данных. При изучении сигналов говорится, что они бывают непрерывными (аналоговыми) или дискретными. Объясняется дискретное представление информации и дискретизация.

Однако непрерывность встречается и в отношении передачи информации. ***Непрерывная (потоковая) передача информации*** означает, что создаётся (открывается) канал связи, по которому затем передаются данные до тех пор, пока не будет закрыт сам канал связи. В этом случае нельзя заранее оценить их объем: приёмник сигнала не может знать размер конечного файла, потому что файла как такового на момент передачи еще не существует. Однако некоторая информация все же есть: при открытии канала задается характеристика в виде типа данных, например, что это аудиопоток, закодированный с использованием определенного кодека. Информация продолжает приниматься до тех пор, пока существует канал связи.

Примером непрерывных данных могут быть ***стриминговые*** (от англ. stream – поток) сервисы, включая аудиостриминг, видеостриминг, текстовый поток.

Задание 1. Определите тип стриминга (аудио-, видеостриминг, текстовый или иной):

- Интернет-радио;
- трансляция на онлайн-телевидении;
- трансляция игры блогером на специальном сервисе;
- подкаст (в режиме реального времени);
- вебинар;
- онлайн-чаты.

Кстати, компьютерные игры на облачных ресурсах являются своеобразным видеостримингом (со стороны провайдера) в сочетании с текстовым стримингом (от игрока). Информация о нажатии клавиш передается на сервер в реальном времени, а с сервера ведется трансляция изображения экрана. При высокой скорости работы сервера, маленькой нагрузке и высокой скорости Интернет, задержка неразличима пользователем, поэтому создается иллюзия игры как на обычном собственном ПК.

С генерацией текстового потока данных мы сталкиваемся также при работе с чат-ботами. Они генерируют ответ в виде текстового потока, который медленно «появляется» на экране. На самом деле мы видим постоянно обновляющийся «слепок» этого потока в виде сохраненного сообщения.

На самом деле потоком данных является и передача изображения с видеокарты на монитор.

Чтобы реализовать обработку потока данных, требуются значительные усилия: знание специальных стандартов, соблюдение определенных правил передачи (с согласованием по времени передачи сигнала, потому что любой компьютер все равно обрабатывает с помощью цифровой подписи (ЦП) дискретные сигналы), но мы можем представить поток данных условно со стороны пользователя как непрерывную передачу информации с неизвестной характеристикой размера. Именно с таким упрощенным представлением мы решаем задачи для подготовки к экзаменам.

Задание 2. Приведите примеры потоковой передачи и тип информации.

Примечание: для классов, работающих по программе «ИТ-вертикаль», можно провести аналогию с коммутацией каналов или коммутацией сообщений при изучении сетей.

Потоки данных могут передавать большие объёмы информации, и, наоборот, большие данные обычно транслируются в формате потока, поскольку требуется их обработка в режиме реального времени. Например, видеопотоки транслируются ото всех уличных камер в городе, собираясь на серверах, между которыми также обмен информацией происходит в формате потоков – так формируются большие данные. Эти данные обрабатываются также в режиме реального времени с помощью различных сложных алгоритмов, например, компьютерного зрения. Затем результат обработки также передается дальше онлайн для оперативного принятия решений людьми.

Более того, необходимость работы с большими данными в потоковом формате следует из самой сущности больших данных (трех основных характеристик – *объёма, скорости и разнообразия*).

Чтобы данные считались большими, должны обязательно выполняться следующие условия, которые считаются их основными характеристиками:

- **объем (Volume)** – огромный объем данных, которые нельзя эффективно обработать с помощью традиционных методов, например, вычисления классических статистических показателей;
- **скорость (Velocity)** – большие данные не только обрабатываются, но и поступают (или генерируются) с огромной скоростью; их необходимо обработать также оперативно в режиме реального времени для принятия решений;

- **разнообразие (Variety)** – большие данные собираются из различных, разнообразных источников, также они могут быть разных типов: от текстов до видео, а еще они бывают структурированные и не структурированные.

Задание 3. *Проверьте, являются ли предложенные примеры данных большими:*

- таблица с именами одноклассников;
- данные городской транспортной системы;
- контент всех пользователей социальной сети.

Для проверки составьте таблицу, отмечая соответствие указанным характеристикам.

Понятия «бинарные деревья» или «дерево решений» – синонимы, не вводятся отдельно, суть понятия разбирается при решении задач по программированию массивов и, соответственно, после изучения графов.

При двоичном поиске упорядоченный массив всегда разбивается на две части. На самом деле аналогичным образом работают **бинарные деревья принятия решений**.

Когда имеющиеся данные представляют в виде графа, похожего на дерево, каждый узел представляет собой условие, то есть вопрос, с помощью которого можно разделить данные текущего шага (порядок узла относительно корня) на части. В случае с бинарными деревьями условие может выполняться или не выполняться. Если оно записано в формате вопроса, на который можно дать ответ «Да» (обычно слева) и «Нет» (ветка продолжается направо). Ветвь дерева – это ребро графа, показывающее выполнение предыдущего условия. Разделение продолжается до тех пор, пока данные возможно разделить на две категории. Если данные уже не делятся, то такой узел является итогом. Но здесь работа алгоритма не заканчивается: нужно собрать решение.

Дерево принятия решений является интеллектуальным алгоритмом, потому что условия (вопросы) формулируются на каждом шаге, исходя из доступных характеристик конкретных данных, а не строго прописываются заранее. Ход принятия решений для разных данных, поданных на вход алгоритма, не совпадают. Важно понимать, что такие алгоритмы только предоставляют помощь в выборе опорных шагов для принятия решений, но само решение всегда за человеком.

Задание 4. Рассмотрим пример дерева решений на рис. 4. В зоомагазине есть пять видов домашних питомцев: морская свинка, канарейка, волнистый попугай, кролик и котёнок. Какое животное выбрать, исходя из пожеланий членов семьи?

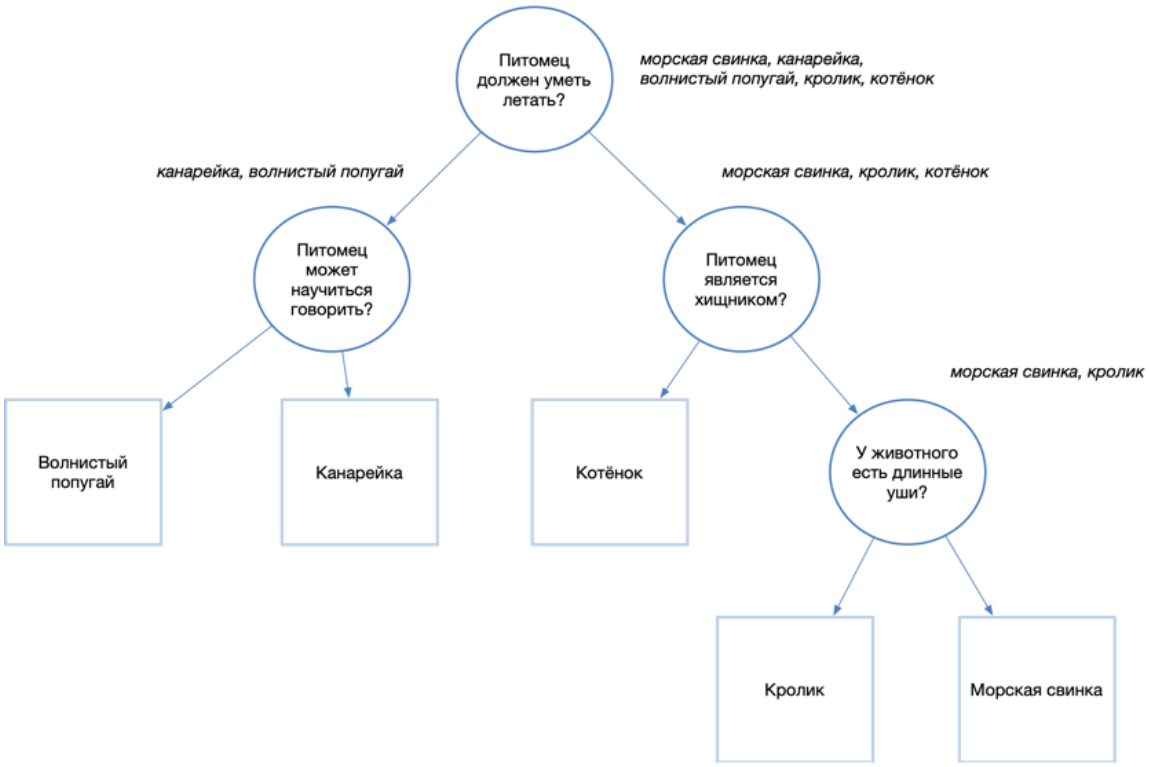

Рис. 4. Дерево решений для выбора домашнего питомца

Итак, морская свинка в качестве питомца подойдёт человеку, который ищет нелетающее животное, которое не является хищником и не имеет больших ушей. Для выбора же канарейки человек должен искать летающего питомца, который не обязан учиться говорить.

Часто в формулировке заданий может запрашиваться, как достигнуть одного конкретного результата. Также данные могут разбиваться по характеристикам на смешанные группы, и тогда к одному и тому же ответу можно будет прийти с помощью разных вопросов. То есть свободные узлы с ответами могут повторяться. Для чего это нужно? Например, в банке человек не хочет или не может сообщить все сведения (или у него нет части документов), но решение о выдаче ипотеки банк принять должен. Тогда рассматриваются все доступные пути решений и выбирается тот, для выполнения которого достаточно данных.

Задание 5. Составьте дерево принятия решений для определения породы собаки: доберман, шпиц, пудель, мопс и бульдог. Обратите внимание, что бульдоги и пудели бывают разных размеров.

В деревьях решений для построения структуры дерева может использоваться рекурсия. С помощью графического представления деревьев удобно иллюстрировать глубину рекурсии.

Задание 6. Попробуем описать пример работы на дереве, решением которого должны стать отдельные числа. Допустим, есть набор чисел: 1, 98, 23, 64, 39, 7. Пусть в узлах происходит сравнение, находится ли число или несколько чисел в первой половине списка по возрастанию величин. Это наше условие (см. рис. 5). При его выполнении идет продолжение работы для левой ветки, а если условие не выполняется, то для правой:

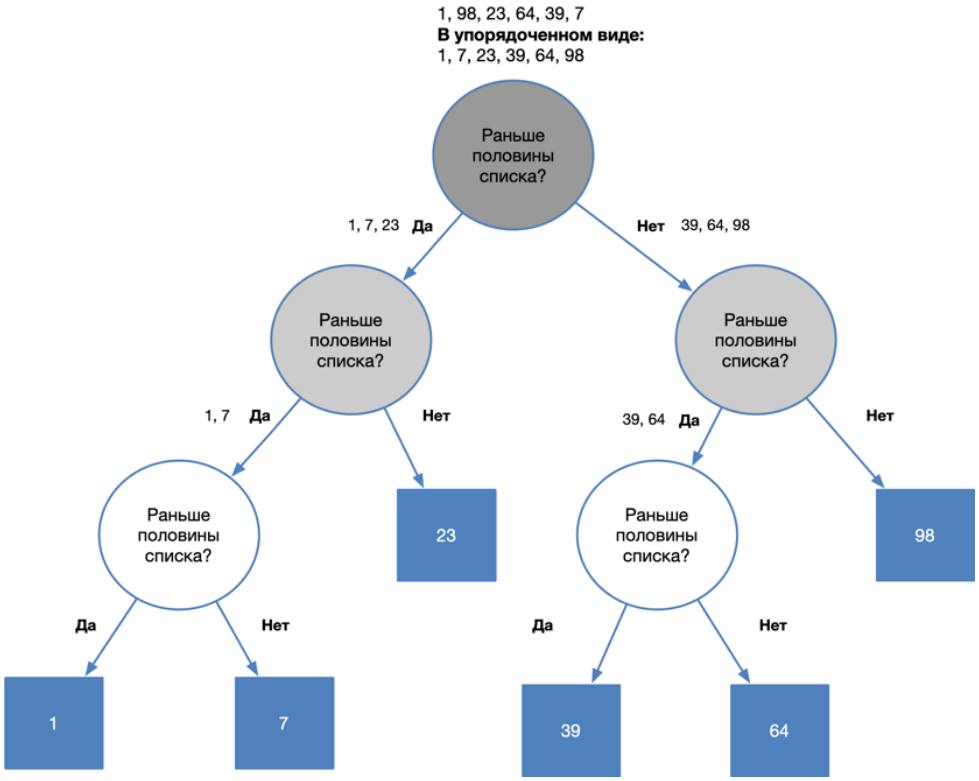

Рис. 5. Дерево решений для числовых данных

Реализуем представленный пример в виде программы с рекурсией:

```
def build_decision_tree(data, level): #Пусть уровни глубины рекурсии начинаются по умолчанию с 1
    if len(data) == 1:  # Если в данных один элемент, то он автоматически становится вершиной-результатом:
        print("Уровень ", level,", достигнут результат:", data)
    # Если остаётся два элемента, то, учитывая, что они уже упорядочены, первый отправляем в левую ветку, второй - в правую:
    elif len(data) == 2:
        left_data = data[0]
        right_data = data[1]
        print("Уровень ", level,"из ", data, ", левая ветка: ", left_data,", правая ветка:", right_data)
    #Во всех остальных случаях требуется выполнить проверку условия в каждом новом узле:
    else:
        left_data = []  # Подготовить пустую левую ветку
        right_data = []  # Подготовить пустую правую ветку
        #Напишем проверку условий в узлах:
        for i in range(len(data)): #Проходим по всем элементам списка
            #Если условие выполняется, то добавляем элемент в левую ветку, иначе в правую:
            if i < len(data) // 2:
                left_data.append(data[i])
            else:
                right_data.append(data[i])
        #Выводим на экран глубину рекурсии и промежуточные результаты на узлах:
        print("Уровень ", level,"из ", data, ", левая ветка: ", left_data,", правая ветка:", right_data)
```

```
        # Вызываем функцию рекурсивно для левой и правой веток, чтобы
продолжить проверку:
        left_tree = build_decision_tree(left_data, level+1)
        right_tree = build_decision_tree(right_data, level+1)


# Пример использования
mas = input("Введите данные через пробел: ").split(' ')
 # Обработаем тип данных, чтобы иметь дело с целыми числами внутри
списка
result = []  # Пустой список для хранения преобразованных значений
for x in mas:  # Проходим по каждому элементу x в списке mas
    result.append(int(x))  # Преобразуем x в целое число и добавляем
его в список result
mas = sorted(result)
print("Исходные данные: ", mas)
# Строим дерево с помощью описанной нами функции
build_decision_tree(mas,1)
```

Ссылка на проект на Mos.Hub:

https://hub.mos.ru/aa.salakhova/ai-codes/-/raw/main/tree1.py

Вывод программы:

Введите данные через пробел: 1 98 23 64 39 7

Исходные данные: [1, 7, 23, 39, 64, 98]

Уровень 1 из [1, 7, 23, 39, 64, 98] , левая ветка: [1, 7, 23] , правая ветка: [39, 64, 98]

Уровень 2 из [1, 7, 23] , левая ветка: [1] , правая ветка: [7, 23]

Уровень 3 , достигнут результат: [1]

Уровень 3 из [7, 23] , левая ветка: 7 , правая ветка: 23

Уровень 2 из [39, 64, 98] , левая ветка: [39] , правая ветка: [64, 98]

Уровень 3 , достигнут результат: [39]

Уровень 3 из [64, 98] , левая ветка: 64 , правая ветка: 98

Здесь рекурсия использовалась для вызова проверки условия при создании новой ветки.

Для дерева принятия решений следует точнее выбирать условия для каждого нового уровня дерева, то есть для новых узлов. Например, условием может быть вопрос, делится ли нацело число на глубину шага (и тут уже условие меняется с каждым уровнем глубины рекурсии). Это

все еще не будет интеллектуальным деревом решений, однако отлично демонстрирует сам принцип построения деревьев.

Примечание для учителя! Рассматриваем готовую программу в качестве демонстрации, учащимся не требуется ее разрабатывать. Также в задании 7 со звездочкой.

Задание 7*. Составьте код для дерева, на узлах которого проверяется делимость числа на номер текущего уровня рекурсии. Ограничьте рекурсию.

```
def build_decision_tree(data, level, max_depth):
    if level == max_depth or len(data) == 0:  # Проверка на
ограничение глубины рекурсии
        return #Остановка выполнения функции
    elif len(data) == 1:  # Если в данных один элемент, то он
автоматически становится вершиной-результатом:
        print("Уровень ", level,", достигнут результат:", data)
        return
    # Если остаётся два элемента, то, учитывая, что они уже
упорядочены, первый отправляем в левую ветку, второй - в правую:
    elif len(data) == 2:
        left_data = data[0]
        right_data = data[1]
        print("Уровень ", level,"из ", data, ", левая ветка: ",
left_data,", правая ветка:", right_data)
    else:
        left_data = []  # Левая ветвь дерева
        right_data = []  # Правая ветвь дерева
        for num in data:
            if num % level == 0:  # Проверка делимости числа на номер
текущего уровня рекурсии
                left_data.append(num)
            else:
                right_data.append(num)
        print("Уровень ", level,"из ", data, ", левая ветка: ",
left_data,", правая ветка:", right_data)

        left_tree = build_decision_tree(left_data, level + 1,
max_depth)  # Рекурсивный вызов для левой ветви
        right_tree = build_decision_tree(right_data, level + 1,
max_depth)  # Рекурсивный вызов для правой ветви


# Пример использования:
mas = input("Введите данные через пробел: ").split(' ')
 # Обработаем тип данных, чтобы иметь дело с целыми числами внутри
списка
result = []  # Пустой список для хранения преобразованных значений
for x in mas:  # Проходим по каждому элементу x в списке mas
    result.append(int(x))  # Преобразуем x в целое число и добавляем
его в список result
mas = sorted(result)
```

```
print("Исходные данные: ", mas)
deep = int(input("Введите глубину рекурсии: "))
# Строим дерево с помощью описанной нами функции
build_decision_tree(mas,2,deep) #с указанием глубины рекурсии, а не
только стартового уровня
```

Страница кода: https://hub.mos.ru/aa.salakhova/ai-codes/-/raw/main/tree2.py

Вывод:

```
Введите данные через пробел: 2 8 9 68 95
Исходные данные: [2, 8, 9, 68, 95]
Введите глубину рекурсии: 5
Уровень 2 из [2, 8, 9, 68, 95] , левая ветка: [2, 8, 68] , правая
ветка: [9, 95]
Уровень 3 из [2, 8, 68] , левая ветка: [] , правая ветка: [2, 8, 68]
Уровень 4 из [2, 8, 68] , левая ветка: [8, 68] , правая ветка: [2]
Уровень 3 из [9, 95] , левая ветка: 9 , правая ветка: 95
```

Мы начали рассматривать со второго уровня, поскольку делимость на 1 проверять нелогично, это никак не влияет на результат.

Внимательно рассмотрите ветви. Иногда на них нет левой части – это не характерно для интеллектуальных бинарных деревьев. В них алгоритм должен сам подобрать условия, которые обязательно разделят выборку чисел на две группы, но вопросы на узлах также не повторяются.

Примеры дополнительных заданий (для домашнего задания и последующего обсуждения в классе, для текущего оценивания усвоения теоретического материала и т.д.).

1. Проверьте, являются ли большими данными:
   - медицинские данные в единой информационной системе;
   - таблица последней поставки продуктов в сельском магазине;
   - данные о товарах в торговой сети.

   Ответ можно оформить в виде таблицы.
2. Составьте дерево принятия решений для определения породы собаки: доберман, шпиц, пудель, мопс и бульдог. Обратите внимание, что бульдоги и пудели бывают разных размеров.
3. Составьте код для дерева, на узлах которого проверяется, есть ли среди введенных букв та, номер которой в алфавите равен значению глубины рекурсии на текущем шаге. Ограничьте рекурсию.

Содержание модуля ИИ в тематическом разделе «Информационные технологии»

Пользователи сети постоянно генерируют новые тексты: иногда сами, иногда с помощью генеративных алгоритмов в составе различных популярных сервисов и чат-ботов.

Генеративные алгоритмы — это алгоритмы машинного обучения, основанные на методах и моделях, позволяющих создавать новые данные из имеющихся с аналогичными характеристиками. Алгоритмы обнаруживают определенные «похожести» внутри набора данных (повторяющиеся символы, сочетания слов и пр.) и оперируют ими при генерации новых данных. Это именно данные, поскольку алгоритмы не извлекают из данных информацию, т.е. смысла здесь нет, только его подобие.

Именно поэтому **базу знаний** – то, на что опираются как справочный материал (и на чем происходит обучение) такие алгоритмы, – очень тщательно отбирают специалисты люди. Также некоторые модели предусматривают поиск в Интернет, например, интеллектуальные поисковые системы, но и их ответ базируется на выявленных при обучении закономерностях.

Генеративные алгоритмы не могут создать ничего по-настоящему нового. Во время их обучения каждый блок обучающих данных разбивается на маленькие наборы (***батчи***), которые затем разбиваются на минимально доступные единицы (***токены***). Далее именно из токенов по выявленным правилам генерируются ответы. Токен не обязательно состоит из одного символа, это может быть как целое слово, так и сочетание нескольких пикселей (по цвету), если речь идёт о генеративных алгоритмах, работающих с изображениями.

Почти все известные и широко применяемые для ежедневных нужд интеллектуальные сервисы работают в виде ***больших языковых моделей*** (LLM, Large Language Model). Это специальные компьютерные модели, которые спроектированы таким образом, чтобы интерпретировать запросы на

естественном языке и предоставлять ответы в таком же формате. Именно так мы взаимодействуем с генеративными чат-ботами. При взаимодействии с такой моделью мы можем описать ожидаемый результат генерации, как если бы мы взаимодействовали с человеком. Первые такие системы появились еще в 1960-х. Также следует понимать, что не каждый ***чат-бот*** (chat bot) на самом деле является интеллектуальным. Это программа, которая имитирует общение с человеком. Если ответы записаны как большой алгоритм с каскадным ветвлением, то чат-бот нельзя будет считать интеллектуальным. Сегодня популярны чат-боты, в которых ответ генерируется на сервере с помощью генеративного алгоритма. Таким образом, все запросы (и введённые вами данные) сначала отправляются на обслуживающий сервер, где и происходит вся «магия».

Генеративные алгоритмы позволяют сгенерировать правдоподобный текст по любой тематике. Он собирается из часто встречающихся сочетаний слов, но сам ИИ не способен понять смысл сгенерированного текста. Компьютер всего лишь имитирует человеческую письменную речь.

После генерации всегда стоит проверять текст:

- на логическую связность и непротиворечивость;
- достоверность указанных фактов и событий;
- корректность названий;
- согласование слов и предложений внутри;
- соответствие контексту;
- согласование стилей изложения в разных абзацах;
- понятность и четкость.

Генеративные алгоритмы «любят» добавлять вводные слова или использовать клишированные метафоры. Как вы думаете, почему так происходит?

Задание 1. С помощью генеративного алгоритма, встроенного в Яндекс.Браузер (YaGPT 2 с Алисой: https://a.ya.ru/), сгенерируйте сказку про любого зверька.

Чтобы генерация произошла правильно, в **промпте** – опорном запросе для *нейросети* – задайте основные требования и ограничения к истории. Например, целевой возраст будущего читателя («для малышей 4-5 лет»), цель генерации («показать важность завтрака»), технические характеристики («на русском языке, не более 2000 символов»), требования к стилю изложения («веселая сказка с описанием от зверька») и обязательные элементы («жилище или домик в лесу, нора или дупло»).

Чем подробнее описан запрос, включающий обязательно ограничения, тем лучше будет происходить генерация. При задании общих и размытых формулировок LLM не может получить достаточно сведений, чтобы ограничить материалы, из которых будет генерироваться ответ, и вы рискуете получить очень странные и некорректные формулировки.

Итак, пример запроса:

«Напиши веселую сказку с описанием от лица бельчонка, показывающую важность завтрака, для малышей 4-5 лет, на русском языке, длиной не более 2000 символов. При описании обязательно используй дупло на самом высоком дубе в лесу».

Задание 2. Нейронные сети могут не только генерировать тексты, но и производить интеллектуальную обработку текста.

Примеры интеллектуальной обработки текста:

- краткий пересказ (выявление ключевых моментов);
- перефраз (пересказ в другом стиле или, например, изменение читательского адреса – того, кто будет потребителем текста);
- форматирование и выделение главного;
- генерация на основе уже готового текста.

Допустим, мы решили, что наша сказка должна быть для ребят 7-8 лет. Снова откроем чат-бот Алисы (https://a.ya.ru/).

Теперь в качестве запроса будет: *«Преобразуй и дополни сказку о бельчонке для возраста 6-7 лет. Добавь негативный пример, что будет, если*

бельчонок не позавтракает перед школой». Затем следует в кавычках разместить оригинальный текст сказки.

Именно способность интеллектуальных систем к определению и анализу контекста дает преимущества **интеллектуальному переводу** над обычными компьютерными программами-переводчиками.

К контексту, который учитывают интеллектуальные переводчики, также относится стиль речи (формальный, неформальный, профессиональный и т.д.), наличие слэнга, жаргонизмов, а также возможность обнаружения фразеологизмов и речевых оборотов внутри предложения, а не дословный перевод. Например, выражение «knock yourself out» вовсе не означает, что вас выгоняют, наоборот, вам предлагают «чувствовать себя как дома». Помимо этого, интеллектуальные переводчики лучше обрабатывают ошибки, которые мог допустить один из участников коммуникации. Современные нейронные переводчики также подбирают примеры контекста использования выражений, что помогает в изучении языков. Зачастую в подобных переводчиках можно дополнительно указать специфические правила преобразования, которые нужны лично вам, например, для аббревиатур.

Нейронные сети, обслуживающие программы интеллектуального перевода, могут помочь распознавать голосовой ввод прямо во время диалога, а также распознавать и переводить надписи в реальном времени при наведении камеры смартфона на объект.

Задание 3. Зайдите на онлайн-сервис интеллектуального переводчика (например, https://translate.yandex.ru/). Переведите вашу сказку на английский язык, чтобы поделиться ею на уроке иностранного языка.

Попробуйте самостоятельно изменить текст, используя функции «Улучшить текст» и сделать его «Сложнее», а затем преобразуйте в «Разговорный стиль».

Задание 4. Давайте познакомимся с другими возможностями интеллектуальной обработки текстов, а именно пересказом и выделением тезисов.

Выделение тезисов может быть полезно, когда вам нужно сделать памятку для повторения материала или для выступления. Будьте аккуратны! Учитывайте необходимость проверки предлагаемого контента.

Перейдите на сайт сервиса быстрого пересказа (https://300.ya.ru).

Для начала сделаем пересказ статьи о белках из «Большой российской энциклопедии»: https://bigenc.ru/c/belki-rod-mlekopitaiushchikh-cca483

Вставьте ссылку в окно ввода. При пересказе будут сформулированы две версии: краткая и подробная. Можно также скопировать ссылку на конкретный пересказ и поделиться ею.

При изучении растровых рисунков и операции редактирования графических объектов можно рассмотреть генерацию растровых изображений. Принцип работы генеративных алгоритмов для генерации изображений похож на работу с текстом. В данном случае базу знаний, состоящую из обучающей выборки изображений, анализируют на зависимости, выявляя закономерности между соотношением различных цветов или их сочетаний (сочетаний пикселей или областей одного цвета) и ***метаданных*** картинки, к которым относится название изображения или контекст, из которого оно взято или как было описано. При запросе от пользователя по зависимостям генерируется основная часть изображения, остальное достраивается с учетом часто встречаемых сочетаний относительно уже заполненных точек картинки.

Поскольку ИИ не понимает смысла, если на этапе обучения добавить множество картинок, например, гусей, с подписью «крокодил», то затем алгоритм будет на запрос «крокодил» генерировать птицу.

Важно понимать, что алгоритмы могут только повторить существующие зависимости и тенденции, но не создавать новые креативные идеи. Так в сообществе digital-художников были даже забастовки, когда создатели нейросетевых сервисов по генерации изображений брали обучающий материал в виде работ художников без их на то разрешения. При использовании чат-ботов, поддерживающих генерацию растровых

изображений, при составлении промпта следует руководствоваться следующими указываемыми характеристиками:

- стиль изображения («в мультяшном стиле», «в стиле киберпанк», «в стиле импрессионизма» и т.д.);
- описание основного объекта (указание ключевых особенностей внешнего вида);
- описание деталей (например, выражение лица для человека либо предмета в руках, описание одежды или окружения, размещенного на том же плане, что основной объект);
- описание переднего плана (если он требуется, «перед героем на столе чашка с кофе»);
- описание заднего плана («на фоне заката»).

Также может потребоваться указать соотношение сторон картинки (например, 1:3) или цветовой гаммы («используй контрастные цвета» и т.д.).

Чем точнее задан запрос, тем более близкий к желаемому будет результат.

Для генерации картинок используйте «Шедеврум» от «Яндекс» (https://shedevrum.ai/text-to-image/).

Задание 6. Сгенерируйте презентацию по мотивам составленной ранее сказки.

Для генерации презентаций воспользуемся российским сервисом PresentSimple (https://presentsimple.ai/). Это проприетарный продукт, однако он предоставляет право на бесплатную пробную генерацию.

В курсе информатики 9 класса в разделе «Информационные технологии» рассматривается тема «Системы умного города (компьютерное зрение и анализ больших данных)», а в разделе «Цифровая грамотность» говорится о знакомстве с сервисами государственных услуг. Поскольку предполагается некоторое обзорное знакомство с указанной тематикой, то лучше их объединить, предварительно подготовив примеры, связанные с регионом проживания. В ходе обзора появятся новые термины: «умный

город», «интернет вещей», «цифровые двойники городов», которые следует обсудить с использованием примеров региона проживания учащихся.

Жители любых городов, не только мегаполисов, сталкиваются с новыми технологиями в контексте централизованной обработки данных и принятия решений. Сложность развернутой инфраструктуры зависит не столько от размера населенного пункта, сколько от уровня информатизации в целом. Например, одним из самых цифровых городов является Иннополис, количество жителей в котором не превышает 500 человек.

Ежесекундно в каждом городе происходит множество вещей, и все они влияют на благополучие и жизнедеятельность горожан. Чтобы поддерживать городскую инфраструктуру для быстрого реагирования всеми службами, создают **цифровые двойники городов**. Это виртуальные модели (системы), показывающие текущее состояние всех подсистем города, анализирующие этот объем больших данных и, зачастую в автоматизированном режиме, регулирующие работу подсистем. Например, система города, получающая информацию о большом количестве осадков и превышении влаги в ливневках на определенной улице, может автоматически сформировать заявку для ремонтной бригады и изменить задержку на светофорах на улицах, а также информировать горожан о трудностях на пути с рекомендацией путей объезда. Конечно, система предлагает несколько решений, а далее их выбирают операторы.

Установленные видеокамеры обеспечивают мониторинг общественных мест (например, станций метрополитена или стадионов во время футбольных матчей) и объектов инфраструктуры для обеспечения безопасности и предотвращения преступлений. Видеокамеры же на перекрестках и дорогах помогают контролировать транспортное движение, распознавать автомобили и оптимизировать управление светофорами для улучшения проходимости и безопасности на дорогах.

Информация собирается не только с камер, но и с других датчиков, а также от горожан через приложения или заявки.

В основном, взаимодействие подсистем умного города происходит в контексте **Интернета вещей (Internet of Things, IoT)**, то есть без прямого участия человека. Различные устройства обмениваются данными по сетям в автоматическом режиме. Не всегда это Интернет в прямом смысле, иногда речь идет о локальной, то есть отдельной, чаще всего закрытой от внешнего вмешательства, сети, созданной специально как компонент умного города. В контексте города используется термин **умный город (Smart City)**.

Системы умных городов требуют значительных ресурсов для их поддержания, в том числе и обслуживания специалистами разных профилей.

Кстати, сами платформы для управления городскими службами, общественной безопасностью, здравоохранением, образованием и другими аспектами жизни в городе, тоже относятся к системе умного города. Например, портал «Госуслуги Санкт-Петербурга» или «Госуслуги Москвы».

Примеры дополнительных заданий (для домашнего задания и последующего обсуждения в классе, для текущего оценивания усвоения теоретического материала и т.д.).

1. Используя информацию про основные виды фишинга, сгенерируйте графические объекты для составления информационного плаката. Оформите текстовый файл плаката в формате А4.
2. Найдите текст статьи о робототехнике на сайте «Большой российской энциклопедии». Оформите краткое сообщение, воспользовавшись сервисами по работе с текстом, включая сокращение и пересказ.
3. Создайте видео для иллюстрации одного из классических произведений, которые вы проходите на уроке литературы сейчас. Для генерации описания сцен используйте приведенные ранее рекомендации.
4. Сгенерируйте презентацию на основе сообщения об истории чат-ботов.
5. Приведите примеры (1-2 на каждый пункт) применения следующего оборудования и технологий в контексте систем умного города:

- видеокамеры,
- ГЛОНАСС (российская спутниковая система навигации),

- облачные технологии,
- датчики протечек,
- call-центр.

Используемая литература:

1. Вариативное обучение основам искусственного интеллекта в общем образовании на основе интегративного подхода: монография / С. Д. Каракозов, Н. Н. Самылкина, А. А. Салахова, Е. А. Самохвалова. – Москва: МПГУ, 2024. – 364 с.: ил.

Приложение 1.
Вклад отечественных ученых в развитие искусственного интеллекта

В России Семён Николаевич Корсаков занимался поиском способа усилить возможности разума с помощью вычислительных машин и механизмов. В 1832 году он изобрёл и опубликовал описание пяти механических устройств, частично механизирующих умственную деятельность при решении задач поиска, сравнения и классификации. В конструкции этих машин впервые в истории информатики были применены перфорированные карты, игравшие роль баз знаний, а сами машины по существу являлись прообразом экспертных систем (ЭС).

В мире есть разные научные школы ИИ, различающиеся, в основном, в подходах и, как следствие, направлениях своего развития. Это также определяет принятое понимание основного определения данной науки у нас. В России традиционно существует единая школа искусственного интеллекта Дмитрия Александровича Поспелова, возникшая в СССР в 1960-х годах в стенах мехмата МГУ.

Русский математик Андрей Андреевич Марков (младший) также внёс большой вклад в алгебру логики. Он создал рабочий аппарат и ввёл понятие **нормальных алгоритмов** (**алгорифмов**). И сегодня мы используем его определение, послужившее прочным фундаментом теории алгоритмов. Важно отметить, что долгое время алгоритмы признавались, только если выдавали одинаковые результаты при нескольких запусках. Однако это несправедливо для большинства интеллектуальных алгоритмов, особенно для нейронных сетей, где результат обучения может отличаться при разных запусках системы.

Важно отметить, что все ученые теоретики опирались на двоичную систему, характерную для алгебры логики. Вычислительный центр МГУ под руководством академика Сергея Львовича Соболева берёт на себя разработку доступного для лабораторий и вузов компьютера. Команда Николая Петровича Брусенцова в 1958 году представляет образец компьютера

«Сетунь», который в 1960 году показал феноменальный результат полезного времени работы — 95%. Особенность данной вычислительной машины заключалась в применении **трёхпозиционной логики**: + («истина»), 0 («неизвестно»), – («ложь»). Подобная логика сегодня широко используется в нейронных сетях как состояния персептрона.

В СССР искусственный интеллект стали изучать одновременно с США. Так в 1954 году в МГУ под руководством выдающегося математика – академика Алексея Андреевича Ляпунова – начал работу семинар «Автоматы и мышление», где зародились два направления работы в сфере ИИ в нашей стране: нейрокибернетики и кибернетики «чёрного ящика». В основном рассматривались вопросы поиска решений логических задач.

В 1960-х годах Вениамин Ноевич Пушкин и Дмитрий Александрович Поспелов возглавили ряд исследований о мыслящих автоматах в Московском университете и Академии наук в рамках области «кибернетика». Советские учёные также опирались на исчисления предикатов и теорию рекурсивных функций. Во многом становление данной области было связано с развитием теории управления, и большинство кибернетиков оставили значительные работы и в этой сфере. Поспелов также является «отцом» нового направления – ситуационного управления.

Сергей Юрьевич Маслов в своей работе 1964 года «Обратный метод установления выводимости в классическом исчислении предикатов» предложил метод автоматического поиска доказательства теорем. Вскоре в Ленинграде на базе Ленинградского отделения математического института им. Стеклова начинает работу группа, занимающаяся доказательством теорем на основе обратного метода резолюций Маслова. Результатом их трудов становится программа АЛПЕВ ЛОМИ.

Через два года Валентин Фёдорович Турчин разрабатывает язык рекурсивных функций РЕФАЛ (РЕкурсивных Функций АЛгоритмический язык), который был популярен ещё долгое время, поскольку позволял работать с задачами перевода с естественного на искусственный язык или же

решать задачи ИИ. Какое-то время РЕФАЛ существует одновременно со сходным по идеологии языком Prolog, разработанным французским учёным Аленом Колмероэ в 1972 году. Однако зарубежная разработка оказалась более лёгкой для изучения студентами и учениками, поэтому в 1980-х годах Prolog был включён в некоторые вузовские и школьные учебники информатики. С его помощью изучались элементы математической логики, основы логического программирования и даже введение в экспертные системы: проектирование баз знаний, создание моделей экспертной системы (ЭС).

Также в СССР ведутся независимые разработки в сфере компьютерного зрения, аналогичные достижениям Розенблатта в США. В 1961 году Михаил Моисеевич Бонгард создаёт алгоритм «Кора», моделирующий деятельность человеческого мозга при распознавании образов, предвосхищая более поздние исследования японского программиста Кунихико Фукусимы.

Официальная история развития ИИ в СССР началась в 1974 году, когда был создан Научный совет по проблеме «Искусственный интеллект», образованный в рамках Комитета по системному анализу при Президиуме АН СССР. Исследовательские центры и проекты дополняются новыми школами для молодых специалистов, проводятся симпозиумы, постоянно работающие семинары, конференции, организуется профильный научный журнал.

До 1974 года в нейронных сетях невозможна была реализация строгой дизъюнкции (XOR, исключающего ИЛИ) и возникали значительные сложности при обучении многослойных сетей, пока Пол Дж. Вербос и Александр Иванович Галушкин (одновременно и независимо друг от друга) не предложили метод обратного распространения ошибки.

Кроме указания на ограниченные способности первых моделей нейронных сетей, в работах Минского содержался фундамент для другого подхода к ИИ – **инженерии знаний (knowledge engineering)**. Также стоит отметить, что отвлечение внимания от нейронных сетей помогло не только

инженерии знаний, но и, например, развитию систем, распознающих естественный язык.

Источники:

1. Искусственный интеллект:10-11 классы: учебное пособие / И.А. Калинин, Н.Н. Самылкина, А.А. Салахова. Москва: Просвещение, – 2023. – 144 с. ил.
2. Обучение основам искусственного интеллекта и анализа данных в курсе информатики на уровне среднего общего образования: монография./ Н.Н. Самылкина, А.А. Салахова// Москва: МПГУ, – 2022. – 228 с. ил.
3. Основы искусственного интеллекта в школьном курсе информатики: история вопроса и направления развития / Н.Н. Самылкина, А.А. Салахова // Информатика в школе. – 2019. – №7(150) сентябрь. – С. 23-32. 1,1 п.л.

Приложение 2.
Организация курса по выбору или внеурочной деятельности для подготовки к олимпиадам по ИИ

Всероссийская олимпиада школьников по информатике (ВсОШ) проводится в четыре этапа (школьный, муниципальный, региональный, заключительный), является доступной для всех организаций общего образования. С 2025 года во ВсОШ по информатике четыре профиля: программирование, искусственный интеллект, информационная безопасность, робототехника. На всех этапах олимпиад по программированию и искусственному интеллекту используется система автоматической проверки заданий, поэтому нет ограничений по количеству участников в каждом этапе, кроме заключительного. Это индивидуальные соревнования для отбора одаренных и подготовленных к программированию учащихся. Сложность заданий нарастает постепенно на каждом этапе. Олимпиады по программированию и искусственному интеллекту больше всего взаимосвязаны между собой по способам подготовки к ним, используемому языку программирования Python, возможностью использовать альтернативную форму поступления в вуз в случае успешности. Поэтому рекомендуется на начальном этапе освоения программирования не делать разделения по этим профилям. Для олимпиады по искусственному интеллекту очень важны основные навыки программирования на Python по темам, которые получат свое развитие в следующих этапах и необходимы для решения задач ИИ. Язык программирования Python пока единственный доступный учащимся язык программирования, для которого есть полный набор средств решения задач искусственного интеллекта.

На базе 7 класса необходимо сформировать навык решения задач на языке Python, с использованием его стандартных функций и библиотек (например, регулярных выражений). У учащихся по завершении обучения можно проверить:

- текстовый ввод и вывод;

- переменные и основные типы данных: int, float, string;
- условия, циклы, перебор;
- классы и объекты: создание, точечная нотация, методы и свойства;
- структуры данных: кортеж, список, словарь;
- модули и пакеты;
- работа с текстовыми файлами;
- регулярные выражения;
- функции и параметры.

Решением задачи считается программный код на языке Python, соответствующий требованиям, описанным в условиях задачи.

Каждому участнику необходимо предоставить возможность проверить работоспособность кода на одном тесте неограниченное количество раз. Затем идет автоматическая проверка на закрытых наборах тестов.

Все решения проверяются набором автоматических тестов, выполняемых в изолированной среде, это позволяет не подгонять код под конкретные выходные данные (как иногда случается на олимпиадах по программированию).

Используемая для подготовки к олимпиадам среда должна включать язык Python с его набором стандартных модулей, а также специализированных библиотек NLTK, pillow, numpy, pandas, scikit-learn, tensorflow (включая keras), PyTorch. Среда не должна иметь доступа к сети Интернет.

Далее (8-9 классы) можно переходить к задачам на использование специализированных библиотек, ориентированных на работу с числовыми, текстовыми и графическими данными, с ориентацией на подготовку и анализ наборов данных для последующего решения задач интеллектуальной обработки. Предусматривается возможность использования библиотек Numpy, Pandas, Pillow или OpenCV, sklearn, MathPlotLib. Задания оценивают умения применить специализированные библиотеки для решения ранее не встречавшейся задачи.

Использовались следующие классы задач:

- классификации;
- кластеризации;
- выявления ассоциативных правил;
- регрессии.

Предполагается, что участники в своем решении:

- анализируют или формируют набор данных для решения задачи;
- выбирают модель для ее решения;
- формируют обучающий и (при необходимости) тестовый набор данных;
- проводят обучение и при необходимости тестирование модели;
- готовят программу, применяющую модель для формирования ответа и выводящую ответ в стандартный поток вывода.

Задачи не должны предусматривать самостоятельной реализации участниками алгоритмов перечисленных классов. Все необходимые для решения модели и средства имеются в составе библиотек, но ими необходимо грамотно воспользоваться.

Таким образом, будущие участники профиля «искусственный интеллект» будут подготовлены к необходимости разработки программной модели обработки данных в соответствии с заданием на заключительном этапе олимпиады по ИИ.

Предлагаем использовать данные учебно-методические материалы учителями информатики для подготовки и проведения курса подготовки к олимпиадам по искусственному интеллекту.

Задачи могут быть решены в любой среде программирования с поддержкой Python. Тем не менее для решения последующих задач, более эффективного исследования и устранения ошибок в программном коде лучше использовать более развитый дистрибутив, ориентированный на работу с языком Python и решение задач Data Science: Anaconda.

В составе дистрибутива есть интегрированная среда разработки Spyder, включающая в себя специализированный текстовый редактор, средства работы с интерпретатором Python, пошагового исполнения, просмотра переменных и другие.

Задача 1

Напишите программу ввода целого числа (от -999999 до 999999) и вывода его текстовой формулировки на русском языке.

Общее описание решения:

Часто встречающаяся задача, в рамках которой можно продемонстрировать средства стандартного ввода и вывода данных, работу с целыми числами и строками. В случае применения языка Python задача позволяет рассмотреть работу со списками, словарями, а также использование функций для структурирования кода.

Решение опирается на правила записи числительных в русском языке: фактически название числа конструируется из количества единиц (числа от 1 до 999), тысяч, миллионов и т.д. В рассматриваемой задаче к этой особенности добавляются также отрицательные числа (их запись предваряется словом «минус»).

Число от 1 до 999 преобразуется в текстовую форму как название количества сотен, десятков и единиц – например, «триста двадцать два». Этот принцип нарушается только для чисел от 11 до 19 – у них есть самостоятельные названия.

Таким образом, для решения задачи мы:

- выделяем знак числа;
- разбиваем число на две части – тысячи и единицы;
- каждую из этих частей записываем словами;
- соединяем части, вставив слово «тысяч», – если тысячи в числе вообще есть. Заметим, что, в зависимости от числа тысяч, у нас может меняться окончание слова – одна тысячА, две (три, четыре) тысячИ и от 6 до 9 тысяч.

Решение задачи:

Подготовим функцию преобразования числа от 1 до 999 в текстовый вид. Принцип работы этой функции довольно прост: мы выделяем единицы, десятки и сотни после чего преобразуем их в слова.

Слова (если есть нужные части) добавим в список, из которого потом и соберем число.

```
def intToText( num = 0, oneName =  'один' , twoName = 'два' ):
# Словарь словарей для выбора диапазона названий, по основанию деления
и числу
    numText = { 1 :  { 0 :'ноль', 1 : oneName ,2 : twoName, 3 :
'три',4 : 'четыре', 5: 'пять', 6 : 'шесть',7 :
'семь',8:'восемь',9:'девять'},
            20 :
{11:'одиннадцать',12:'двенадцать',13:'тринадцать',14:'четырнадцать',15
:'пятнадцать',16:'шестнадцать',17:'семнадцать',18:'восемнадцать',19:'д
евятнадцать'},
            10 : {1 : 'десять', 2 : 'двадцать',3 : 'тридцать',4 :
'сорок',5 : 'пятьдесят',6:'шестьдесят', 7 : 'семьдесят',8 :
'восемьдесят',9: 'девяносто'},
            100 : {1 : 'сто',2 : 'двести',3 : 'триста',4 :
'четыреста',5 : 'пятьсот',6 : 'шестьсот', 7:'семьсот', 8:'восемьсот',
9:'девятьсот'} }

    result = [] # Список частей, изначально пустой
    if num //100 in numText[100]: # Если есть сотни
        result.append(numText[100][num//100]) # Добавляем название
количества сотен
        num = num % 100 # Добавляем название количества сотен
    if num //10 in numText[10]: # Если есть десятки
        if num //10 == 1 and num %10 != 0: # Если остаток – от 11 до
19
            result.append(numText[20][num]) # Добавляем название от 11
до 19
            num = -1 # В этом случае преобразование закончено
        else:
            result.append(numText[10][num//10]) # Десятков больше 1,
добавляем название
            num = num % 10 # выделяем единицы
    if num >=0 in numText[1]: # Если единицы есть
        result.append(numText[1][num])# Добавляем название единиц
    return ' '.join(result) # Собираем название из списка – соединяем
их в строку через пробел
```

Единственной особенностью этой функции являются два дополнительных параметра – название чисел 1 и 2 – для правильного использования в названии тысяч. По умолчанию это обычные названия для чисел от 1 до 999.

Для определения правильного склонения слова «тысяча» по количеству тысяч, напишем несложную вспомогательную функцию:

```
def getEnd1000 ( num = 1):
    num = num %100
    if num %100 >9 and num %100 <=20:
        return 'тысяч'
    else:
        num = num %10
    if num == 1:
        return 'тысяча'
    if num >= 2 and num <=4:
        return 'тысячи'
    return 'тысяч'
```

Теперь можно написать и основную программу:

```
digit = - 1000000
while digit < -999999 or digit > 999999: # Вводим число –
соответствующее условиям
    digit = int( input('Введите целое число от -999999 до 999999:') )

sign = -1 if digit < 0 else 1 # Определяем и сохраняем знак
thousand =  abs(digit) // 1000 # Количество тысяч
entities = abs(digit) % 1000 # Количество единиц до 1000

print ( ('минус ' if sign == -1 else '' )+(intToText( thousand,
oneName =  'одна' , twoName = 'две' )+' '+getEnd1000(thousand) +' ' if
thousand >0 else '') + intToText(entities) ) # Собираем строку,
учитывая знак и количество тысяч (есть ли они)
```

Задача 2

Напишите программу ввода текстового представления целого числа (от -999999 до 999999) и перевод этого числа в целочисленную форму.

Общее описание решения:

Задача, обратная к предыдущей задаче – что позволяет использовать их для проверки друг друга. Решение строится на тех же особенностях записи чисел, которые были рассмотрены в первой задаче: число нужно разбить на группы по три разряда. Каждую такую группу со значением от 0 до 999 домножить на сдвиг разрядов (если он есть – например, на тысячу или миллион) и результаты сложить. Внутри группы запись чисел также разбивается на названия и полученные числа складываются (разрядность чисел уже учтена в названии).

Для решения задачи:

- Введем число – точнее, его словесное представление. (На полную правильность во время ввода проверять не удобно, но такую проверку можно сделать в самом преобразовании).
- Выделим слово «минус», чтобы определить «отрицательность» числа.
- Разобьем введенное число уже без «минус» по слову «тысяча» (учитывая варианты написания).
- Каждую полученную (до «тысяча» и после «тысяча») часть преобразуем отдельно: для преобразования тройки разрядов в число разобьем строку на слова, каждое слово преобразуем в число, результаты сложим.
- Преобразованное до слова «тысяча» слева – домножим на тысячу, справа – просто прибавим к первой части.

Решение задачи:

Воспользуемся для операций со строками механизмом регулярных выражений.

```
import re
```

Чтобы избежать дублирования кода и структурировать решение, подготовим функцию преобразования строки с записью трехразрядного числа в числовое представление:

```
def strDigitalToNum (  strDigital = 'ноль' ):
    wordDigital = {'сто' : 100 , 'двести' : 200 , 'триста' : 300
,'четыреста' : 400 ,
                   'пятьсот' : 500,'шестьсот' : 600, 'семьсот': 700,
'восемьсот' : 800, 'девятьсот' : 900, 'десять' : 10, 'двадцать' : 20,
'тридцать' : 30, 'сорок' : 40, 'пятьдесят':50, 'шестьдесят':60,
'семьдесят' : 70, 'восемьдесят' : 80,'девяносто':90, 'одиннадцать' :
11, 'двенадцать' : 12,'тринадцать' : 13, 'четырнадцать' : 14,
'пятнадцать' : 15, 'шестнадцать' : 16, 'семнадцать' : 17 ,
'восемнадцать':18,'девятнадцать' : 19, 'ноль' : 0,  'один' :1, 'одна'
:1  , 'два' :2, 'две' : 2 , 'три' : 3, 'четыре':4, 'пять' : 5, 'шесть'
: 6 ,'семь' : 7,'восемь':8,'девять':9 }
    digitalNum = 0
    for digitalStr in re.split(r'[ ]+',strDigital):
        digitalNum += wordDigital[digitalStr]
    return digitalNum
```

Как видим, большая часть этой функции – словарь, в котором слову сопоставлено число. Накапливаем результат: изначально число равно нулю, после чего мы перебираем строки, которые нам возвращает разбиение по регулярному выражению. Выражение очень простое – элементы в строке разделяются пробелами (возможно, их между словами больше одного).

Каждое слово используем как ключ словаря и получаем соответствующее ему число.

Основная программа тоже не велика:

```
strDigital = input('Введите словесное представление числа:').lower()
negativeIs = re.split(r'минус[ ]+',strDigital)# Разбиваем по строке
«минус »
negative = len(negativeIs) – 1# Если минус был – результат 1, если нет
- 0
strDigitalSep = re.split(r'[ ]+тысяч[аи]?[ ]+', negativeIs[negative])

if len(strDigitalSep) == 2:
    res =
strDigitalToNum(strDigitalSep[1])+strDigitalToNum(strDigitalSep[0])*10
00
else:
    res = strDigitalToNum(strDigitalSep[0])

res *= (-1 if negative==1 else 1)

print( f'Запись в числовой форме: {res}' )
```

В выделении тысяч мы используем регулярное выражение, учитывающее возможные окончания: «а» и «и» могут быть, а может не быть ничего.

Если тысячи выделились – преобразовываем обе части, если нет – только единицы. После этого остается только учесть знак и вывести результат.

Задача 3

Даны два текстовых потока (для проверки это могут быть файлы, но размер их вам заранее неизвестен и может считаться бесконечным). Из потоков поступают возрастающие натуральные числа. Выдать в третий поток объединение чисел из входных потоков, сохранив принцип возрастания.

Общее описание решения:

Особенность этой задачи – отсутствие возможности прочитать все значения в память и выполнить сортировку. Тем не менее, поскольку мы уверены в соблюдении условия внутри потоков, алгоритм решения этой задачи достаточно очевиден:

- читаем по одному значению из файлов (который фактически может быть потоком данных);
- если первое значение больше – записываем его и считываем новое из первого файла;
- иначе записываем второе значение и считываем его из второго файла;
- повторяем с шага 2, пока оба файла не закончатся.

Обратите внимание:

- мы не можем каждый раз читать два значения сразу,
- никто не обещает, что потоки равной длины (да и вообще, что они не пусты с самого начала).

Для решения задачи:

Для решения задачи на Python нам придется учесть несколько особенностей:

1. В языке Python нет циклов с проверкой условия в конце.
2. В языке Python нет отдельной функции, проверяющей достижение конца файла.
3. Поскольку мы не можем считать ни один файл в память целиком, использование привычного способа чтения файлов – перебором строк с помощью итераторов – будет невыгодным.
4. Строка читается с символом конца строки – то есть длина строки, которую удалось прочесть из файла, минимум 1 символ (‘\n’).
5. Если достигнут конец файла, то функция чтения строки вернет пустую строку.

6. Поэтому достижение конца файла мы будем проверять через длину прочитанной строки.

Решение задачи:

```
stream1 = open ('stream1.txt','r') # Открываем файлы на чтение
stream2 = open ('stream2.txt','r')
stream3 = open ( 'stream3.txt','w' ) # Открываем файл на запись

stream1Text = stream1.readline() # Читаем первое значение из первого
потока
stream2Text = stream2.readline()# Читаем первое значение из второго
потока
while len(stream1Text) > 0 or len(stream2Text)>0 : # Продолжаем
считывать, пока хотя-бы в одном потоке есть данные

    if len(stream1Text) >0: # Если есть значение из первого потока
        if len(stream2Text) == 0 or int( stream1Text ) < int(
stream2Text ):
# Второй поток уже пуст или значение из него больше первого
            stream3.write( f'{int(stream1Text)}\n' )
            stream1Text = stream1.readline()
        else:
           # Может, просто второе значение меньше?
            if len(stream2Text) >0 :
                stream3.write( f'{int(stream2Text)}\n' )
                stream2Text = stream2.readline()
    else:
           # Первый поток кончился, но возможно есть второй?
        if len(stream2Text) >0 :
            stream3.write( f'{int(stream2Text)}\n' )
            stream2Text = stream2.readline()

stream1.close()
stream2.close()
stream3.close()
```

Несмотря на всю простоту решения классической задачи слияния двух отсортированных последовательностей, его реализация требует аккуратности в рассмотрении различных случаев и написании условий, а также учета особенностей языка.

Задача 4

Поступает файл с текстом литературного произведения (больше 50Кб) на русском языке. Файл имеет одну из следующих кодировок: UTF-8 (с BOM), CP1251, Cp866, KOI8R, Macos, ISO 8859-5. Определите кодировку файла без внешних библиотек.

Описание решения:

Полного и точного решения для задачи определения кодировки текста, как мы знаем, не существует. Но мы можем, опираясь на известные технические и статистические данные, подготовить программу, которая будет это делать с достаточной долей вероятности в основной массе случаев.

С учетом того, что список кодировок нам известен и объем текста достаточно велик, мы можем воспользоваться такой эвристикой:

Текст в кодировке UTF-8 имеет обязательное начало: **Byte Order Mark,** то есть два первых байта будут FF и FE (символ U+FEFF).

Поскольку текст достаточно длинный и на русском языке, будут соблюдаться общие статистические закономерности – то есть примерно одни и те же буквы будут встречаться чаще других.

Таким образом, алгоритм будет таким:

- Проверяем два первых символа. Если они соответствуют BOM – этот текст в кодировке UTF-8.
- Подсчитываем количество каждого символа в файле.
- Сортируем символы по убыванию их количества.
- Сравниваем первые три символа с кодами заданных кодировках.

Решение задачи:

```
sourceName = input("Filename=>")# Получаем имя файла
source = open(sourceName,'rb') # Читаем его как поток бинарных данных
freq = {} # Словарь для подсчета количества каждого значения байта в
файле
code1 = source.read(1)
code2 = source.read(1)

if code1 == 255 and code2 == 254: # BOM ?
    print('UTF-8')
      exit(0)
# Учитываем символы в подсчете, раз это однобайтовая кодировка
c = int.from_bytes(code1,"big")
if c > 127:
    freq[c] = 1

c = int.from_bytes(code2,"big")
if c > 127:
    freq[c] = 1
# Читаем файл по одному байту и учитываем количество каждого значения
while True:
    code = source.read(1)
    if not code:
        break
```

```
    c = int.from_bytes(code,"big")
    if c > 127:
       if c in freq:
           freq[c] = freq[c]+1
       else:
           freq[c] = 1
source.close()
# Сортируем словарь по ключу
sortedFreq =  sorted(freq, key=freq.get)
# Реверсируем, чтобы получить нужный порядок
sortedFreq.reverse()

# Создаем словарь
resort = {}
for w in sortedFreq:
    resort[w] = freq[w]

detectTable = [
(0xAE,0xA5,0xA0),(0xEE,0xE5,0xE0),(0xCF,0xC5,0xC1),(238,229,224),
(0xDE,0xD5,0xD0)  ]
detectResult = [0,0,0,0,0]
codeTables = ['CP866','CP1251','KOI8R','Macos','ISO 8859-5']
detectCode = list( resort.items())
# Подсчитываем - какой кодировке больше соответствует порядок?
position = 0
maxPosition = 0
for code in detectTable:
    if code[0] == detectCode[0][0]:
        detectResult[position] += 1
    if code[1] == detectCode[1][0]:
        detectResult[position] += 1
    if code[2] == detectCode[2][0]:
        detectResult[position] += 1
    if detectResult[position] > detectResult[maxPosition]:
        maxPosition = position
    position += 1

print(codeTables[maxPosition])
```

Задача 5

В текстовом файле описаны промежутки на числовой шкале (действительных чисел). Каждый промежуток – два числа в скобках, разделенные точкой с запятой. Если число включается в промежуток – скобка к нему примыкает квадратная, если не включается – круглая. Написать программу, которая «свернет» весь набор промежутков до минимально необходимого, упорядоченного по расположению промежутков на числовой оси (промежутки из файла могут пересекаться, промежутки в результате объединения – нет).

Общее описание решения:

Решение задачи фактически сводится к прибавлению нового промежутка (диапазона) к существующему списку. При этом нам нужно учитывать, что промежуток может пересекаться (накладываться) с уже имеющимися промежутками, и тогда нужно будет изменить их границы – то есть промежуток может оказаться внутри имеющегося, может наложиться на него частично (и тогда границу нужно отодвинуть), может оказаться на самом краю – и тогда нужно будет учесть положение граничных точек.

Промежутки мы считываем из файла по одному, разбираем и превращаем в множество из 4 элементов: левый край (число), признак вхождения края в диапазон (Да/Нет), правый край (число) и признак вхождения правого края. Левый край промежутка всегда меньше правого.

При появлении нового промежутка мы проверяем все, уже имеющиеся, и вырабатываем новый список диапазонов.

Таким образом, алгоритм будет такой:

- создаем пустой список диапазонов;
- открываем файл для чтения;
- считываем строку и создаем диапазон;
- проверяем имеющийся список поэлементно;
- если новый диапазон не пересекается с элементом – добавляем элемент в новый список;
- если пересекается – расширяем новый диапазон;
- добавляем новый диапазон со всеми изменениями в список диапазонов;
- повторяем с пункта 3 пока не исчерпаем файл.

Решение задачи:

Создадим несколько вспомогательных функций:

```
def intersect ( a, b) : # Пересекаются ли промежутки A и B?
    if a[0] <= b[0]:
        if b[0] < a[2] or (b[0] == a[2] and a[3] and b[1] ) or (b[0]
== a[0] and a[1] and b[1]):
            return True
```

```
        else:
            return False
    else:
        if a[0] < b[2] or (a[0] == b[2] and b[3] and a[1] ):
            return True
        else:
            return False
```

Промежутки пересекаются, если хотя-бы одна точка одного оказывается внутри другого – нам только нужно учесть, что крайняя точка может входить, а может не входить в диапазон.

Функция слияния промежутков:

```
def getUnion ( aR, bR ): # Слияние промежутков
    a = min( aR[0], bR[0] )
    b = max( aR[2], bR[2] )
# Если края равны и хотя-бы один входит
    if aR[0] == bR[0] and ( aR[1] or bR[1] ):
        aIn = True
    else:
# Края не равны, так что признак как у выбранного
        aIn = aR[1] if aR[0] == a else bR[3]
    if aR[2] == bR[2] and ( aR[3] or bR[3] ):
        bIn = True
    else:
        bIn = aR[3] if aR[2] == b else bR[3]
    newRange = (a,aIn,b,bIn)
    return newRange
```

Слияние промежутков – это определение его новых границ. Определить границу не сложно – выбираем минимальный из левых промежутков и максимальный из правых. Нужно учесть, что после этого придется определить – входит ли граница в диапазон?

Теперь можно написать функцию добавления диапазона – фактически, пересоздавая список:

```
def addRange( ranges = [], newRange = ()):
    newRanges = []
    for currentRange in ranges:
        if not intersect( currentRange , newRange ) :
# Промежутки не пересекаются – добавляем проверяемый
            newRanges.append( currentRange )
        else:
# Промежутки пересекаются – изменяем проверяемый
            newRange = getUnion( currentRange , newRange )
# В итоге добавляем получившийся новый диапазон – после всех проверок
    newRanges.append(  newRange )
    return newRanges
```

Основная программа с использованием этих функций достаточно проста:

```
ranges = []
with open('ranges.txt','r') as rangesFile:
    for newRangeStr in rangesFile:
        newRangeList = newRangeStr.replace('\n', '').split(',')
        newRange = ( float(newRangeList[0][1:]), False if
newRangeList[0][0]=='(' else True,
                     float(newRangeList[1][:-1]), False if
newRangeList[1][-1]==')' else True
                     )
        ranges = addRange( ranges,  newRange )

ranges.sort( key = lambda x : x[0])

for currentRange in ranges:
    print (f'{"[" if currentRange[1] else
"("}{currentRange[0]},{currentRange[2]}{"]" if currentRange[3] else
")"}')
```

Для создания диапазона мы считываем строку и делим ее на части регулярным выражением. Каждый диапазон добавляем в список, по окончании чтения сортируем его и выводим на экран.

Для решения более сложных задач необходима не только среда разработки, но и дополнительные библиотеки. В составе дистрибутива Anaconda есть большая часть этих библиотек, но, если их все-таки не окажется, их можно будет установить стандартными средствами Python.

Задача 6

Постройте график функции $f(x) = -1^{[|x|]}x$ *для x из [-10;10]. Квадратные скобки в показателе степени означают целую часть числа.*

Общее описание решения:

Для решения задачи построения графика мы, конечно, применяем специализированную библиотеку mathplotlib. График строится на основе двух массивов данных библиотеки Numpy.

С формальной точки зрения все просто:

- заполняем массив значениями x с достаточной плотностью;
- вычисляем значения функции (формула не сложная);
- строим график.

Однако, если мы построим график таким образом, результат будет неверным – потому, что mathplotlib построит линию, соединив все точки. Из-за показателя степени в ненулевых целых точках функция терпит разрывы – у нее меняется знак, что приводит к красивому, но неверному результату.

Поэтому для построения оценочного графика придется воспользоваться механизмом маскировки – в составе Numpy это специализированный класс массивов, которые позволяют исключить отдельные значения.

Для решения задачи:

- заполняем массив значениями оси абсцисс (x);
- вычисляем точки маскировки;
- маскируем эти значения;
- вычисляем значения ординат (создаем массив с нужными значениями);
- строим график, опираясь на эти массивы;
- добавляем точки, исключенные из основного графика (см. рис. 6).

Решение задачи:

```
import matplotlib.pyplot as plt
import numpy as np

t = np.arange(-10, 10.0, 0.01) # Заполняем массив абсцисс
powt = np.absolute(np.modf(t)[1]) # Отдельно вычисляем их целые части

tf = np.arange(-10, 10.0, 1) # Вычисляем точки разрыва

tm = np.ma.masked_values(t, tf[0] ) # Маскируем точки разрыва, начиная
с -10
for a in tf:
    tm = np.ma.masked_values(tm, a )# Используем функцию для сравнения
чисел с плавающей точкой!

y = ((-1)**powt) * tm # Создаем массив ординат - выполняя операции над
массивами Numpy
ytf = ((-1)**tf) * tf # Отдельно посчитаем исключенные точки

fig, ax = plt.subplots() # Создаем полотно для построения
ax.plot(tm, y) # Строим график
ax.scatter(tf, ytf, s=5, color = 'r') # Добавляем к диаграмме
исключенные точки

ax.set(xlabel='X', ylabel='Y', title='$y = -1^{[|x|]}x$') # Добавляем
подписи
ax.grid() # Добавляем сетку
```

```
fig.savefig("test.png") # Сохраняем результат
plt.show() # Показываем, если позволяет среда
```

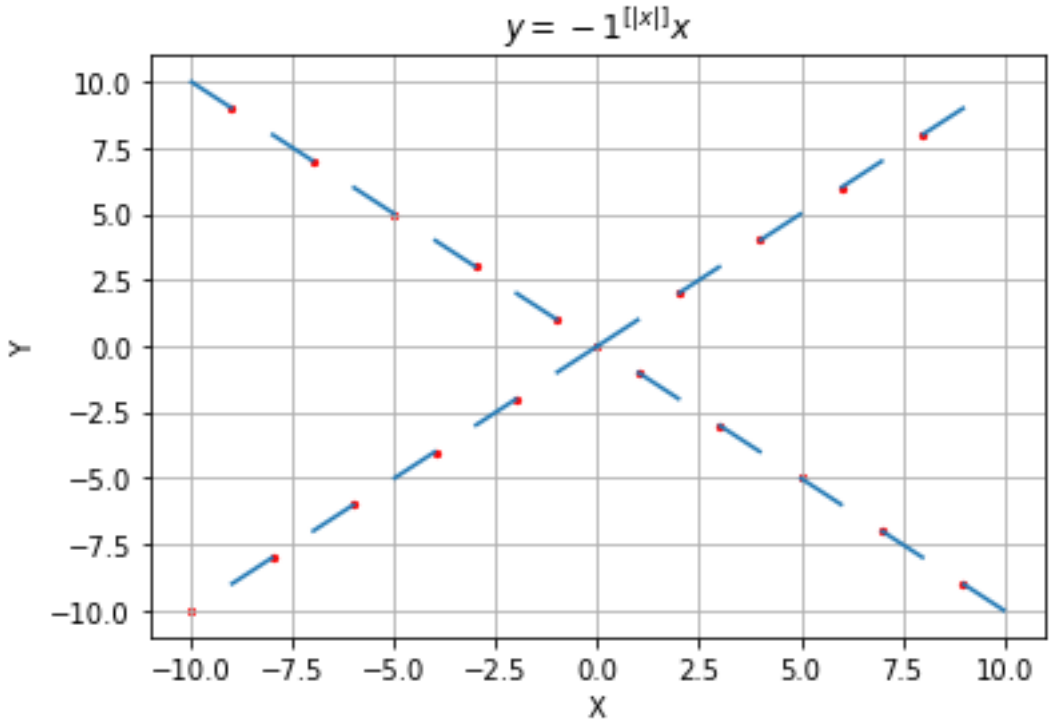

Рис. 6. График функции

Точки разрывов обозначены отдельно только для того, чтобы подчеркнуть: функция там есть, но продолжается с другой стороны оси (если не выделить – разницы между точками визуально не будет).

Задача 7

Постройте диаграмму встречаемости в порядке убывания 20 самых частых слов из текстового файла (объемом до 50Мб) в кодировке UTF-8. Текст набран по правилам машинописного набора, возможны переносы слов.

Общее описание решения:

Решение задачи кажется довольно очевидным.

Мы должны:

- прочесть текстовый файл построчно;
- выделить из каждой строки слова;
- подсчитать количество раз, которое встречается каждое слово;
- выбрать 20 наиболее частых слов;

- построить диаграмму.

Но нам придется учесть некоторые технические особенности, в частности:

- ✓ Слово может быть разделено на две строки (переносы).
- ✓ Слова могут быть записаны с большой и маленькой буквы (например, если слово стоит первым в предложении).
- ✓ Слова могут разделяться рядом символов, причем несколько разделителей могут идти подряд.
- ✓ Некоторые слова пишутся через дефис – и это одно слово.

Для устранения переносов (поскольку текст большой, но все-таки его объем ограничен и может быть размещен в памяти) мы соединим все строки и удалим фрагменты по шаблону, соответствующему переносу – то есть непробельный символ, тире и символ «конец строки», а после еще непробельные символы.

Весь полученный текст преобразуем к нижнему регистру.

Деление на слова мы выполним с помощью регулярного выражения, где учтем и разделители, и слова через дефис.

Для решения задачи:

- запросим имя файла и откроем его на чтение;
- считаем все строки и объединим их;
- удалим переносы;
- поделим его на слова с помощью регулярного выражения;
- перечислим все слова, и для подсчета слов используем словарь: если слова нет в словаре, добавим как одно вхождение, если есть – увеличим количество вхождений.

Решение задачи:

```
import re

textFile = input("Введите имя файла для анализа=>")
words = {}
wordSplit = re.compile(r"([А-Яа-я]+)([-]{1}[А-Яа-я]+)?")

with open(textFile,'r',encoding='utf-8') as f:
```

```
    strings = f.readlines()
    text = ' '.join(strings)
    text = re.sub(r"([^\s])-\n\s{1}",'\g<1>',text)
    text = text.lower()
    for wordG in re.findall(wordSplit,text):
        word = ''.join(wordG)
        if word in words:
            words[word] = words[word] + 1
        else:
            words[word] = 1

# Отсортируем словарь и отберем первые 20
sortedDict = sorted(words.items(), key=lambda item:
item[1],reverse=True)[0:20]
# Разделим полученный список пар на два отдельных ряда – подписи и
значения
x, y = zip(*sortedDict)
# Построим столбчатую диаграмму
plt.bar(x,y)
# Слова выведем вертикально, иначе они накладываются
plt.xticks(range(len(x)), x, rotation='vertical')
plt.show()
```

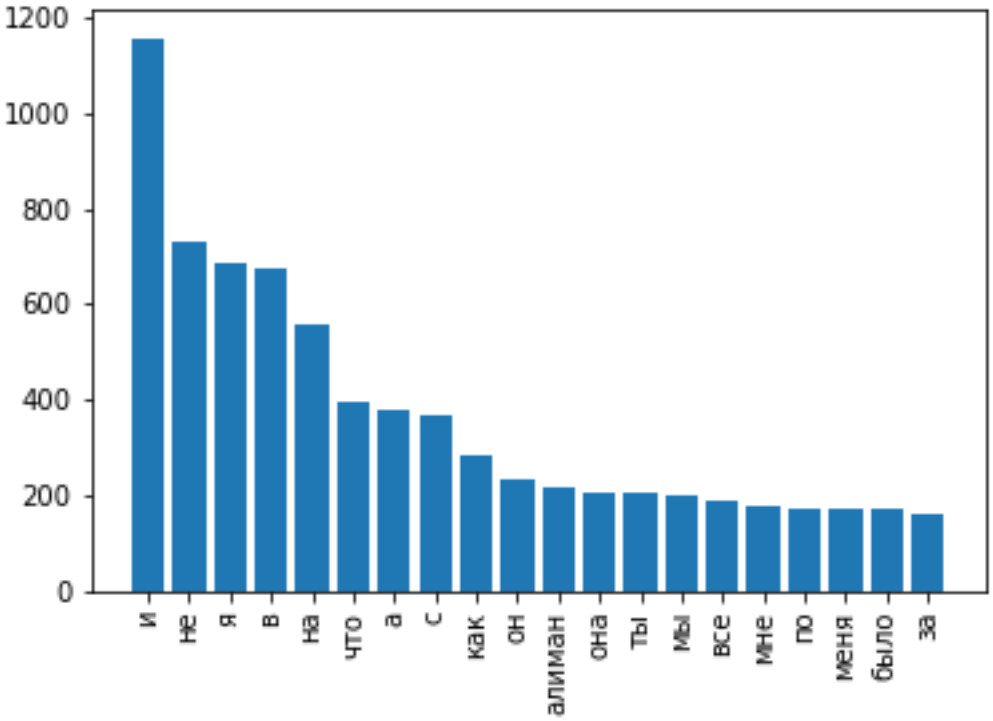

Рис. 7. Диаграмма встречаемости

Задача 8

Вам предоставили две фотографии в формате JPEG: фотографию кота (единственный крупный объект на фото, контрастный, фон светлый) и другого помещения. Сделайте фотографию-коллаж, то есть разместите фото кота на новом фоне. Программ редактирования изображений в вашем

распоряжении нет, есть только Python с установленной библиотекой Pillow и программа просмотра графических файлов.

Общее описание решения:

Библиотека Pillow предназначена для работы с изображениями в растровом формате. Таким образом, изображение, которое мы прочитаем, будет набором точек, – и где именно там кот, как он выглядит, мы заранее знать не можем. Тем более у нас нет возможности точно описать контур кота.

Для решения таких задач в рамках библиотеки у нас есть инструмент «Маска», который позволяет нам определить меру прозрачности каждой точки на изображении. Инструмент применяется при наложении двух изображений. Если точка полностью прозрачна – при наложении маски ее будет не видно (то есть будет видна точка другого изображения). Если полностью непрозрачна – то основного.

Общий алгоритм, который мы применяли бы, используя графический редактор, таков:

- подбираем разницу, которая позволяет нам выделить фон (на основании того, что фон светлый и кот от него заметно отличается);
- создаем новую картинку, на которой точки кота – белые, а точки фона – черные;
- размываем эту картинку (чтобы не возникло резкой границы между котом и новым фоном);
- накладываем кота с маской на новый фон.

Для решения задачи:

- загружаем файл с изображением кота;
- подбираем границу и создаем черно-белое изображение (на котором выделится фон);
- очищаем изображение от мелких артефактов – например, когда в шерсти кота есть почти белые шерстинки;
- загружаем новый фон;
- вставляем на него изображение кота с подготовленной маской.

Решение задачи:

```
from PIL import Image, ImageFilter, ImageOps

filename_cat = "cat3.jpg"

with Image.open(filename_cat) as img_cat:
    img_cat.load()

threshold = 210 # Пороговое значение
# Создаем заготовку маски – отмечаем черным все светлые точки
step_1 = img_cat.point(lambda x: 0 if x > threshold else 255)
# Конвертируем изображение в черно/белое без оттенков
step_1= step_1.convert("1")

# Шаг 2 – убираем мелкие в 1-2 точки артефакты
# Первый фильтр – убирает мелкие белые точки, выбирает минимальный
уровень
step_2 = step_1.filter(ImageFilter.MinFilter(3)
# Второй фильтр – убирает мелкие черные точки, выбирает максимальный
уровень
cat_mask = step_2.filter(ImageFilter.MaxFilter(3))
# Полученную маску конвертируем в градации серого
cat_mask = cat_mask.convert("L")
# Размываем изображение – оно изменится только на границах черного и
белого
cat_mask = cat_mask.filter(ImageFilter.BoxBlur(10))

# Загружаем новый фон
filename_monastery = "monastery.jpg"
with Image.open(filename_monastery) as img_monastery:
    img_monastery.load()

# Вставляем на новый фон изображение, уменьшив его до 20% исходного
размера
# с наложением подготовленной маски
img_monastery.paste(
    img_cat.resize((img_cat.width // 5, img_cat.height // 5)),
    (1300, 750),
    cat_mask.resize((cat_mask.width // 5, cat_mask.height // 5)),
)

# Показываем результат
img_monastery.show()
```

Результат получится примерно такой, как на рис. 8:

Рис. 8. Результат решения задачи 8

Для подготовки к заключительному этапу олимпиады по искусственному интеллекту могут понадобиться следующие специализированные библиотеки:

- SciKitLearn
- TensorFlow / PyTorch
- Средства визуализации

Предлагается рассмотреть одну задачу машинного зрения.

Задача 9. *Программист Никодим любит программировать дома, обедая бутербродами. Бутерброды стоят на отдельном столе.*

Кот Бит программиста Никодима тоже любит бутерброды.

Когда Никодим погружается в любимую работу, он не успевает вовремя сказать: «Брысь!» — своему коту, и часто остается без бутербродов. Он обзывает кота «Мегабайтом», но все равно огорчается.

Наблюдать за котом и бутербродами Никодим не может, потому что он работает. Говорить Биту: «Брысь!» — просто так (когда кот просто

ходит вдоль стола или сидит на табуретке) — нельзя, потому что кот тогда перестанет на это реагировать.

Никодим установил старый ноутбук с камерой для наблюдения за бутербродами, записал свой голос, но ему все равно приходится отвлекаться от работы, чтобы нажать на кнопку, — и кот успевает (за 4 секунды) стащить колбасу.

Поскольку искусственным интеллектом Никодим не занимается, он просит написать ему программу, которая будет получать фото стола с бутербродами и выдавать слово «Брысь!», если кот уже лезет на стол не более, чем за три секунды. Если кота нет (или он не лезет на стол), программа должна выдавать прочерк: «-----».

Возможны следующие комбинации:

1. В кадре стол (а на столе тарелка с бутербродами), надо реагировать на кота на столе. Вокруг стола темный фон, камера на нем кота практически не позволяет различить. Коту на столе нечего делать.

2. В кадре не только стол, но и табуретка (иногда с котом), поэтому надо реагировать, только если край кота приближается к бутербродам. Бутерброды лежат на тарелке, тарелка круглая (но фото делается сбоку) и стоит на одном и том же месте. Никодим обещает фото-образец.

3. Все то же самое, что и в пункте два — но тарелку Никодим может ставить в разные места. Никодим обещает только не двигать стол и ноутбук.

За каждый пункт начисляется 10 очков. На реакцию у программы есть 3 секунды — иначе кот успевает стащить колбасу.

Общее описание решения:

Основная трудность задачи – отсутствие точного шаблона «кот» и способа его поиска. Именно такая задача относится к задачам машинного зрения, то есть решается средствами искусственного интеллекта.

С практической точки зрения это задача выделения (то есть сегментации) объектов на фотографии и их последующей классификации. Самостоятельно подготовить и обучить нейронную сеть для таких целей – задача трудная и требующая больших ресурсов, но мы можем воспользоваться уже обученной сетью.

Одна из наиболее развитых и эффективных свободно доступных сетей – семейство сетей YOLO, на момент решения этой задачи была доступна 8 версия сетей, в частности, сеть сегментации и классификации изображений, предобученная на наборе изображений COCO.

Стоит отметить, что для работы это семейство сетей потребует загрузки довольно большого объема дополнительных средств и библиотек (например, PyTorch) – общий объем будет около 1Гб.

При наличии такого средства, задача в первом случае может быть решена так:

- загрузить и подготовить к работе сеть;
- получить изображение;
- выполнить сегментацию и классификацию объектов, отбросив все сегменты с вероятностью менее 0.5 (то есть маловероятные);
- если среди найденных сегментов есть сегмент класса “cat” – на столе кот, – надо реагировать.

Решение части 1:

```
from ultralytics import YOLO
import cv2

# Загружаем модель среднего размера для сегментации
# При первом обращении она будет загружаться с сайта разработчика
model = YOLO("yolov8m-seg.pt")
# Загружаем названия классов объектов
yolo_classes = list(model.names.values())
classes_ids = [yolo_classes.index(clas) for clas in yolo_classes]
# Минимальная вероятность распознавания
conf = 0.5

# Будем продолжать работу пока будут вводить имена файлов
cont = True

while cont:
    file = input('Введите имя файла:')
```

```
    cont = (file != '') # Если ввели пустую строку, прекращаем работу
    if cont:
        img = cv2.imread(file) # Читаем изображение
        hasCat = False # Есть кошка? Пока не нашли…
# Сегментируем изображение и класссифицируем найденное
        results = model.predict(img, conf=conf)
        for result in results: # Перебираем все результаты
            for mask, box in zip(result.masks.xy, result.boxes):
              # Проверяем, не кошка ли нашлась?
                 hasCat = hasCat or (result.names[ int(box.cls[0]) ] ==
'cat')
        print( 'Брысь!' if hasCat else '----') # Кот в кадре есть.
Выгнать.
```

Решение части 2:

Вторая задача часть несколько сложнее. Кот может присутствовать в кадре, но нам надо оценить расстояние от него до тарелки. Оптимальный способ – найти центр объекта «тарелка» (во втором варианте это можно сделать заранее, возможно, даже вручную) и определить расстояние от крайней точки кота до этого центра.

Недостатком такого решения будет то, что если тарелка стоит ближе к краю стола, то такая окружность будет захватывать лишнее – например, кусок стены.

Поэтому более универсальным (хотя и трудоемким) решением будет описать контур границы многоугольником – то есть цепочкой координат. При таком подходе расстояние не вычисляется, а определяется – входит ли точка кота в многоугольник?

Для предварительной разметки – описания контуров – можно установить и использовать свободно доступное программное обеспечение Label Studio (https://labelstud.io/).

Система разметки устанавливается в соответствии с инструкциями на сайте, как приложение Python, и доступна через браузер на машине пользователя. Создадим новый проект для разметки фотографий:

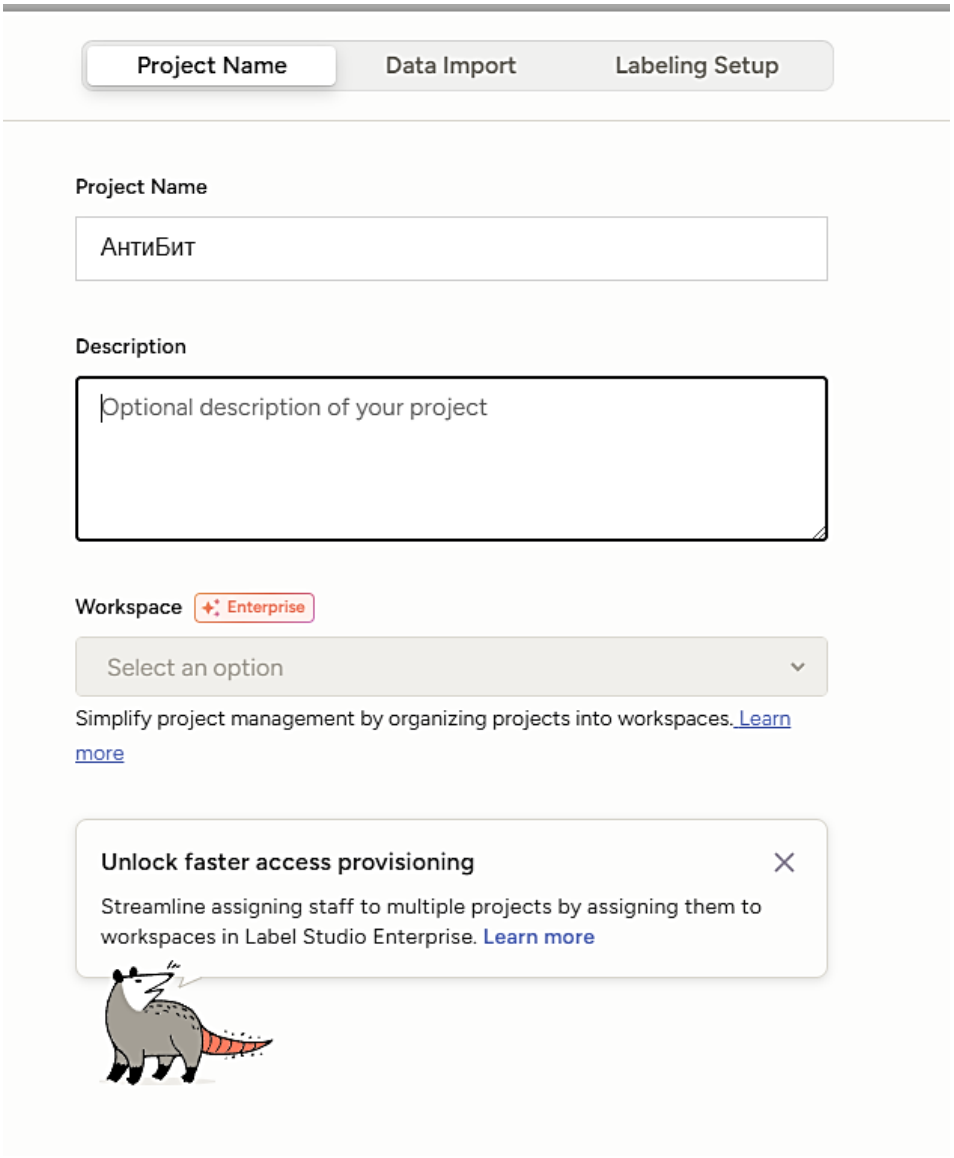

Рис. 9. Создание нового проекта

И загрузим туда фотографию стола с тарелкой (рис. 10):

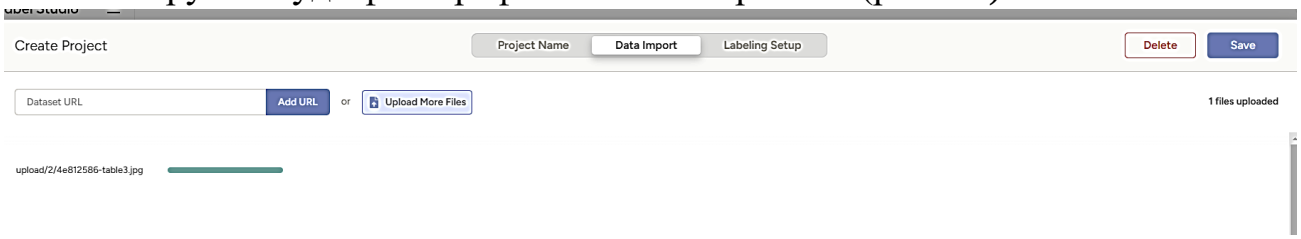

Рис. 10. Загрузка фотографии

Выберем задачи компьютерного зрения, семантическую разметку многоугольниками (см. рис. 11):

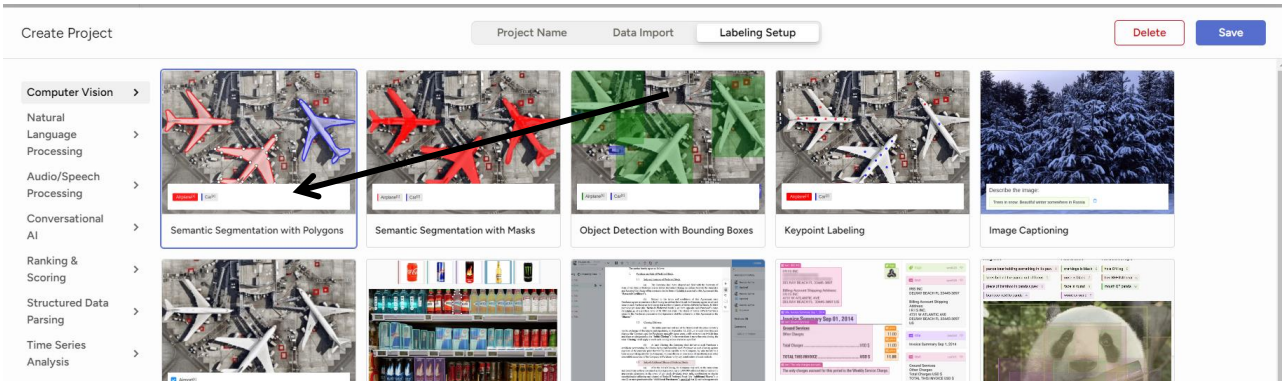

Рис. 11. Выбор задачи и разметки

Зададим категории разметки: Dinning Table и Bowl (именно такие есть в списке категорий Yolo) (см. рис. 12).

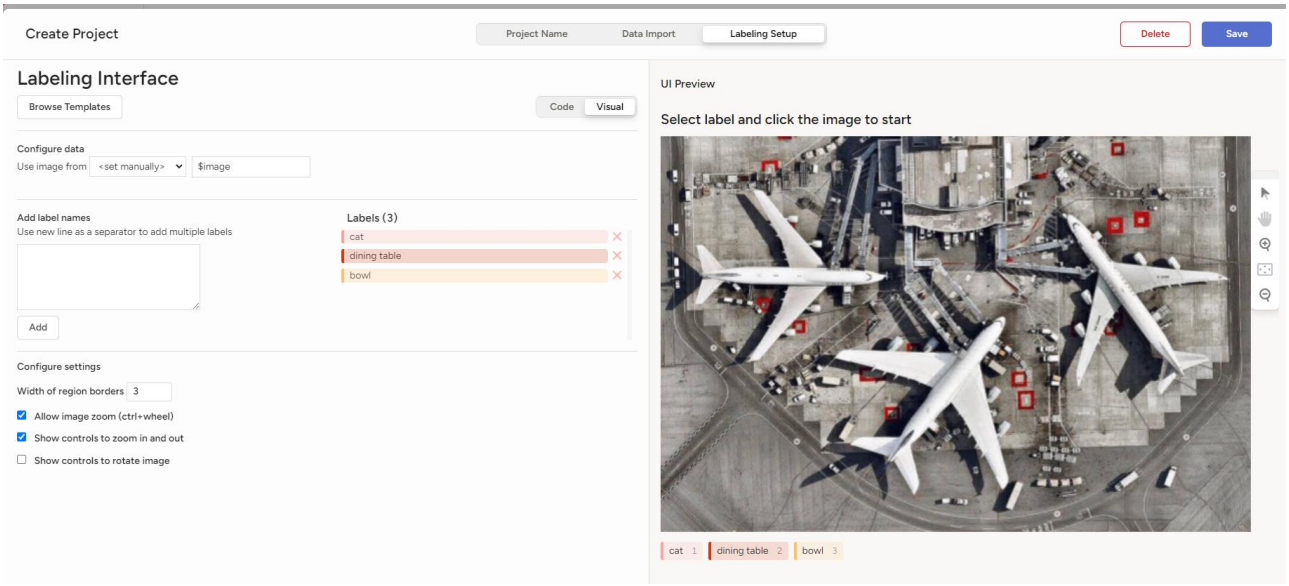

Рис. 12. Выбор категории разметки

Результат сохраним. Теперь откроем фотографию, выберем категорию и поставим точки по контуру объекта. Последнюю точку совместим с первой и получим размеченный объект – тарелку (см. рис. 13).

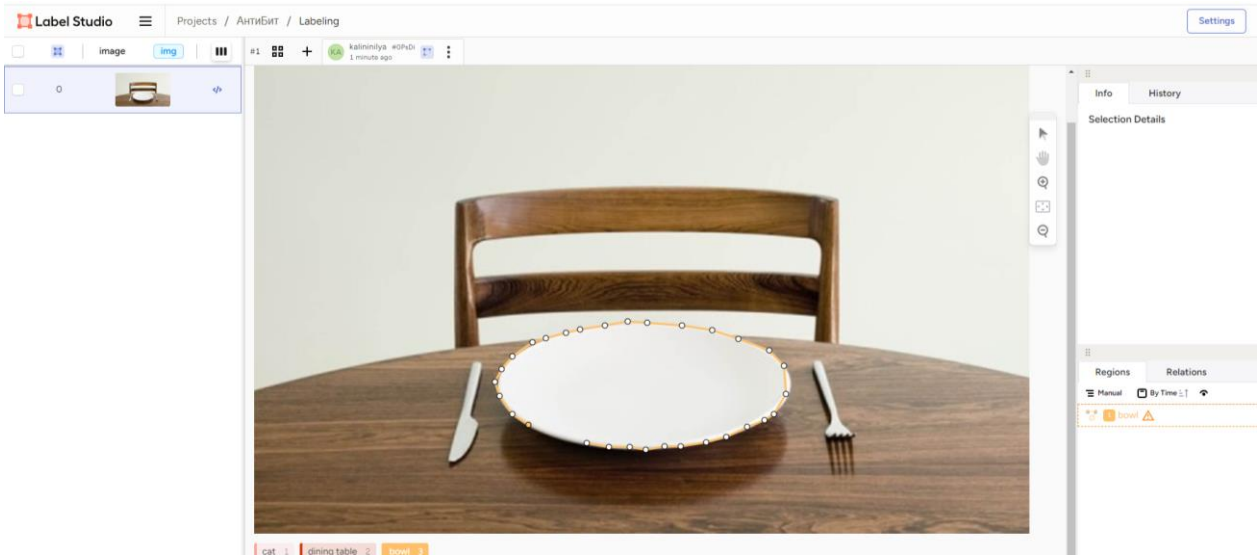

Рис. 13. Разметка сегмента для тарелки

Аналогично выделим стол.

Размеченную фотографию мы экспортируем в формате YOLO (см. рис. 14 и 15).

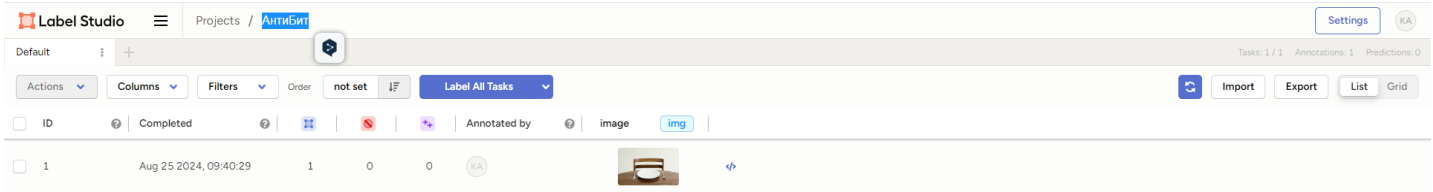

Рис. 14. Экспорт фото

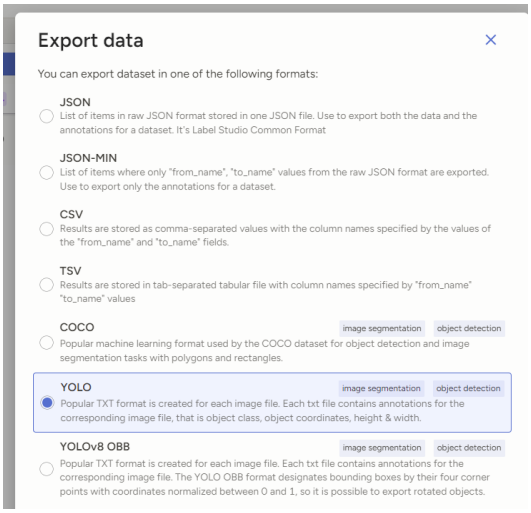

Рис. 15. Выбор формата YOLO

Формат предназначен для обучения, но мы в созданном каталоге найдем файл с координатами и импортируем его в нашу программу. Координаты в

файле записываются подряд (X1,Y1,X2,Y2 и т.д.), так что мы преобразуем результат в словарь списков координат:

```
borders = {
'bowl' : [
(0.34948096885813146,0.7715327784531937),(0.33044982698961933,0.750490
9754044702),(0.3157439446366782,0.7182268773964275),(0.312283737024221
4,0.6831572056485551),(0.3209342560553633,0.6565042551201721),(0.33477
50865051903,0.6298513045917891),(0.3607266435986159,0.6003927803235761
),(0.3780276816608996,0.5919760591040868),(0.40397923875432523,0.58075
37641447676),(0.4221453287197232,0.5737398297951931),(0.45415224913494
81,0.5653231085757037),(0.4835640138408304,0.5569063873562143),(0.5086
505190311419,0.5597119610960442),(0.5536332179930795,0.565323108575703
7),(0.592560553633218,0.575142616665108),(0.6262975778546713,0.5933788
459740017),(0.6660899653979239,0.6172262227625549),(0.6851211072664362
,0.6494903207705975),(0.6974314612722924,0.6847758058830721),(0.692906
5743944638,0.712615729916768),(0.6721453287197232,0.7504909754044702),
(0.6600346020761246,0.7631160572337042),(0.6375432525951557,0.77714392
59328532),(0.610726643598616,0.798185728981576 7),(0.5847750865051904,0
.8094080239408961),(0.5519031141868512,0.8178247451603853),(0.53114186
85121108,0.8178247451603853),(0.5069204152249135,0.8248386795099598),(
0.4861591695501 7296,0.8192275320303001),(0.4593425605536332,0.81922753
20303001),(0.43079584775086505,0.8108108108108109),(0.4048442906574395
4,0.8023940895913213),(0.384083044982699,0.8023940895913213),(0.365051
9031141869,0.7897690077620874)],
'dinning table' : [
(0.02076124567474045,0.6941924623585524),(0.05017301038062279 4,0.67876
18067894886),(0.07958477506505 15,0.670345085569999),(0.11764705882352
938,0.657720003740765),(0.1375432525951557,0.6479004956513608),(0.1548
442906574394,0.6422893481717011),(0.1929065743944636 3,0.63667820069204
15),(0.2404844290657439,0.6226503319928925),(0.2647058823529411,0.6100
252501636585),(0.3019031141868512,0.6058168895539138),(0.3382352941176
47,0.6002057420742543),(0.3581314878892733,0.5931918077246797),(0.3693
7716262975767,0.5959973814645095),(0.4931137546009786,0.58751423663869
),(0.6280276816608996,0.5903862339848499),(0.6513840830449825,0.591789
0208547648),(0.6729215119027362,0.5949640717920244),(0.689446366782006
8,0.5974001683344244),(0.7309688581314877,0.604414102683999),(0.777681
660899654,0.6086224632937436),(0.8451557093425605,0.6212475451229776),
(0.8806228373702422,0.6352754138221266),(0.930795847750865,0.649303282
5212757),(0.9653979238754324,0.6619283643505098),(0.9809688581314877,0
.6689422987000841),(0.9991349480968857,0.6759562330496587),(1.0,1.0),(
0.0008650519031141468,1.0),(0.0008650519031141468,0.7040119704479566)]
}
```

Обратите внимание, координаты заданы не в пикселях (потому что размеры распознаваемых картинок могут быть очень разными), а в десятичных дробях от 0 до 1.

Алгоритм работы мы уже описали, но есть особенность – если мы будем использовать готовые средства определения внутренней точки (в составе SymPy – библиотеки символьной математики для Python), то время работы программы для одной фотографии превысит 5 секунд, что неприемлемо.

Таким образом, нам понадобится своя реализация алгоритма определения внутренней точки многоугольника (причем форма его может быть любой). Самый распространённый алгоритм решения такой задачи – трассировка луча (подсчет пересечений из нашей точки), но этот алгоритм имеет целый ряд особенностей, из-за которых плохо работает для сложных многоугольников.

Воспользуемся более универсальным (хотя и более медленным) алгоритмом суммирования углов.

```
def angleToLine( p , a,b ): # Угол из точки на отрезок
    ax,ay = a[0] - p[0], a[1] - p[1]
    bx,by = b[0] - p[0], b[1] - p[1]

    if  (ax == 0.0 and ay == 0.0) or ( bx == 0.0 and by == 0.0 ):
        return 0 # Точка на вершине

    len_a, len_b = math.sqrt( ax*ax+ay*ay ), math.sqrt( bx*bx+by*by )
    scalar_prod = ax*bx+ay*by
    cos_gamma = scalar_prod/( len_a * len_b ) # Косинус угла

    return ( -1 if (ax*by-ay*bx) <0 else 1) * round(math.acos(
cos_gamma ),10) # Угол, с учетом знака векторного произведения

def innerPoint ( p, poly ):

    angle = 0
    lenP = len(poly)
    for i in range( lenP ):
        angle += angleToLine(p , poly[i] , poly[ 0 if i== (lenP-1)
else i+1 ])
    print(angle)
    return abs(round(angle/math.pi,6)) == 2 # Если угол 2Pi - точка
внутри
```

Таким образом, наша программа выглядит примерно так:

```
from ultralytics import YOLO
import math
from PIL import Image

def angleToLine( p , a,b ): # Угол из точки на отрезок
    ax,ay = a[0] - p[0], a[1] - p[1]
    bx,by = b[0] - p[0], b[1] - p[1]

    if  (ax == 0.0 and ay == 0.0) or ( bx == 0.0 and by == 0.0 ):
        return 0 # Точка на вершине

    len_a, len_b = math.sqrt( ax*ax+ay*ay ), math.sqrt( bx*bx+by*by )
    scalar_prod = ax*bx+ay*by
    cos_gamma = scalar_prod/( len_a * len_b ) # Косинус угла
```

```
    return ( -1 if (ax*by-ay*bx) <0 else 1) * round(math.acos(
cos_gamma ),10) # Угол, с учетом знака векторного произведения

def innerPoint ( p, poly ):

    angle = 0
    lenP = len(poly)
    for i in range( lenP ):
        angle += angleToLine(p , poly[i] , poly[ 0 if i== (lenP-1)
else i+1 ])
    print(angle)
    return abs(round(angle/math.pi,6)) == 2 # Если угол 2Pi - точка
внутри

borders = {
'bowl' : [
(0.34948096885813146,0.7715327784531937),(0.33044982698961933,0.750490
9754044702),(0.3157439446366782,0.7182268773964275),(0.312283737024221
4,0.6831572056485551),(0.3209342560553633,0.6565042551201721),(0.33477
50865051903,0.6298513045917891),(0.3607266435986159,0.6003927803235761
),(0.3780276816608996,0.5919760591040868),
(0.40397923875432523,0.5807537641447676),(0.4221453287197232,0.5737398
297951931),(0.4541522491349481,0.5653231085757037),(0.4835640138408304
,0.5569063873562143),(0.5086505190311419,0.5597119610960442),(0.553633
2179930795,0.5653231085757037),(0.592560553633218,0.575142616665108),(
0.6262975778546713,0.5933788459740017),(0.6660899653979239,0.617226222
7625549),(0.6851211072664362,0.6494903207705975),(0.6974314612722924,0
.6847758058830721),(0.6929065743944638,0.712615729916768),(0.672145328
7197232,0.7504909754044702),(0.6600346020761246,0.7631160572337042),(0
.6375432525951557,0.7771439259328532),(0.610726643598616,0.798185728981
5767),(0.5847750865051904,0.8094080239408961),(0.5519031141868512,0.8
178247451603853),(0.5311418685121108,0.8178247451603853),(0.5069204152
249135,0.8248386795099598),(0.48615916955017296,0.8192275320303001),(0
.4593425605536332,0.8192275320303001),(0.43079584775086505,0.810810810
8108109),(0.40484429065743954,0.8023940895913213),(0.384083044982699,0
.8023940895913213),(0.3650519031141869,0.7897690077620874)],
'dinning table' : [
(0.02076124567474045,0.6941924623585524),(0.050173010380622794,0.67876
18067894886),
(0.07958477508650515,0.670345085569999),(0.11764705882352938,0.6577200
03740765),(0.1375432525951557,0.6479004956513608),(0.1548442906574394,
0.6422893481717011),(0.1929065743944636 3,0.6366782006920415),(0.240484
4290657439,0.6226503319928925),(0.2647058823529411,0.6100252501636585)
,(0.3019031141868512,0.6058168895539138),(0.338235294117647,0.60020574
20742543),(0.3581314878892733,0.5931918077246797),(0.36937716262975767
,0.5959973814645095),(0.4931137546009786,0.58751423663869),(0.62802768
16608996,0.5903862339848499),(0.6513840830449825,0.5917890208547648),(
0.6729215119027362,0.5949640717920244),(0.6894463667820068,0.597400168
3344244),(0.7309688581314877,0.604414102683999),(0.777681660899654,0.6
086224632937436),(0.8451557093425605,0.6212475451229776),(0.8806228373
702422,0.6352754138221266),(0.930795847750865,0.6493032825212757),(0.9
653979238754324,0.6619283643505098),(0.9809688581314877,0.668942298700
0841),(0.9991349480968857,0.6759562330496587),(1.0,1.0),(0.00086505190
31141468,1.0),(0.0008650519031141468,0.7040119704479566)]
}

# Загружаем модель среднего размера для сегментации
```

```
# При первом обращении она будет загружаться с сайта разработчика
model = YOLO("yolov8m-seg.pt")
# Загружаем названия классов объектов
yolo_classes = list(model.names.values())
classes_ids = [yolo_classes.index(clas) for clas in yolo_classes]
# Минимальная вероятность распознавания
conf = 0.5

# Будем продолжать работу пока будут вводить имена файлов
cont = True
while cont:
    file = input('Введите имя файла:')
    cont = (file != '') # Если ввели пустую строку, прекращаем работу
    if cont:
        img = Image.open(file) # Читаем изображение
        xS,yS = img.size
        # Запретные зоны
        alarmZone = {}
        for zone in borders:
            # пересчитать зоны под файл
            alarmZone[zone] =  [ (int(b[0]*xS),int(b[1]*yS)) for b in
borders[zone]  ]
        hasCat = False # Есть кошка? Пока не нашли…
# Сегментируем изображение и класссифицируем найденное
        results = model.predict(img, conf=conf)
        for result in results: # Перебираем все результаты
            for mask, box in zip(result.masks.xy, result.boxes):
              # Проверяем, не кошка ли нашлась?
               if result.names[ int(box.cls[0]) ] == 'cat':
                   alarm = False # Надо выгонять эту кошку?
                   for point in mask:
                       point = (int(point[0]),int(point[1]))
                       for zone in alarmZone: # Кошка в зон тревоги?..
                           alarm = alarm or innerPoint( point,
alarmZone[zone] )
               hasCat = hasCat or alarm # Какая-то кошка в зоне
тревоги?
            if hasCat :
                break
        print( 'Брысь!' if hasCat else '----') # Выгнать, если кот в
запретной зоне.
```

Предложенное решение позволяет нам перейти и к третьей части задачи.

Ее сложность в том, что нужно не просто воспользоваться готовыми контурами, а найти сначала контур стола и тарелки.

При этом стандартная сеть далеко не всегда распознает на фотографиях тарелку (“bowl”) и стол (dinner table), особенно в ситуации, когда от них виден только кусок. Сеть необходимо дообучить.

Дообучение сети выполняется отдельной программой, и для этого потребуется:

- Подобрать (или сделать) серию фотографий (скорее всего, хватит 25–30 штук) стола и тарелки с бутербродами.
- Разметить эти фотографии (то есть вручную максимально точно определить границы объектов) и указать новые классы объектов на ней в уже рассмотренной нами Label Studio.
- Экспортировать данные и подготовить программу дообучения типовой сети (Решение подобной задачи описано, например, в статье «Обучите YOLOv8 на пользовательском наборе данных»[4]).
- Провести обучение – в том числе, подобрать мета-параметры и выбрать количество эпох обучения.
- Сохранить результат.
- В программе распознавания выбрать не готовые контуры, а отобрать контуры объектов нужных классов. Пользуясь тем, что стол стоит на месте, его контур можно и зафиксировать.
- При появлении «кота» на фото мы просто проверим – попадают ли точки «кота» в объект «стол» и «тарелка».

Используемая литература:

Григорьев С. Г., Калинин И. А., Самылкина Н. Н. Система заданий для первой всероссийской олимпиады школьников по искусственному интеллекту // Информатика и образование. – 2022. № 3. – С. 12-20. DOI: 10.32517/0234-0453-2022-37-3-12-20

[4] Статья «Обучите YOLOv8 на пользовательском наборе данных». 2023. URL: https://habr.com/ru/articles/714232/ (дата обращения: 15.08. 2024).