РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ
им. А. И. ГЕРЦЕНА

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

КАФЕДРА ИНФОРМАЦИОННЫХ И КОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

ДИПЛОМНАЯ РАБОТА

Разработка приложения “Умный музыкальный плеер”

Студентки 4 курса
Бражкиной Анастасии Дмитриевны

Руководитель: к.п.н., доцент
Атаян Ануш Михайловна

Санкт-Петербург
2023 г.

Содержание

Введение

В современную цифровую эпоху музыка играет неотъемлемую роль в нашей повседневной жизни. С ростом популярности музыкальных сервисов и повсеместным использованием стриминговых платформ в интернете спрос на музыку как никогда высок. Однако пользователям становится все труднее найти приложение для проигрывания музыки оффлайн, полностью соответствующее их ожиданиям.

Для решения этой проблемы **актуальной является задача** разработки умного музыкального плеера, который будет прост и удобен в использовании, а также функционален и интересен. Основные функции, которые могут заинтересовать пользователя, помимо стандартных функций, заложенных в любой музыкальный проигрыватель, – это генерация обложен к песням с помощью искусственного интеллекта и распознавание слов песни.

Актуальность задачи подтверждается Указом Президента РФ от 10.10.2019 N 490 "О развитии искусственного интеллекта в Российской Федерации" (вместе с "Национальной стратегией развития искусственного интеллекта на период до 2030 года").

Предмет работы – разработка умного музыкального плеера.

Теоретическая значимость работы заключается в исследовании применения методов машинного обучения для генерации изображений и распознавания речи и внесении вклада в растущий объем исследований в области искусственного интеллекта.

Практическая значимость данной дипломной работы заключается в создании музыкального плеера, который может обеспечить необычный опыт использования, генерируя изображения по названиям песен и распознавая текст любой композиции.

Целью данной работы является разработка прототипа умного музыкального плеера, который может воспроизводить музыкальные композиции, генерировать к ним уникальные изображения, распознавать слова. Плеер должен быть реализован в виде десктоп-приложения с использованием языка программирования Python, использование которого обеспечит его эффективность и возможность интеграции с различными платформами и устройствами.

Для достижения цели были поставлены следующие **задачи**:

1. Проанализировать знания о предметной области “машинное обучение” и

существующие музыкальные проигрыватели.
2. Выбрать инструменты для разработки умного музыкального плеера.
3. Разработать технические требования к умному музыкальному плееру и его интерфейс.
4. Разработать прототип умного музыкального плеера.

Структура дипломной работы обусловлена её предметом, целью и задачами: введение раскрывает актуальность, предмет, цель, задачи, теоретическую и практическую значимость работы. В первой главе представлен анализ предметной области и обзор программ-аналогов. Во второй – описание проектирования умного музыкального плеера и реализации его прототипа. В заключении рассматриваются результаты и необходимая будущая работа, включая потенциальные улучшения.

Глава 1. Анализ предметной области и методы машинного обучения

1.1 Анализ предметной области

1.1.1 Машинное обучение

Машинное обучение - это область искусственного интеллекта (Рисунок 1), которая фокусируется на разработке алгоритмов и моделей, позволяющих компьютерам учиться и делать прогнозы или решения без явного программирования.

Рисунок 1 – Области Искусственного Интеллекта

Машинное обучение заключается в создании математических моделей, которые могут анализировать и интерпретировать большие объемы данных с целью выявления закономерностей, взаимосвязей и тенденций. Эти модели обучаются на исторических или помеченных данных, служащих примерами для алгоритма, из которого можно извлечь уроки. Цель состоит в том, чтобы обобщить модель на основе обучающих данных и сделать точные прогнозы или решения относительно новых, невидимых данных.

Алгоритм в машинном обучении, также называемый моделью – это математическое выражение, представляющее данные в контексте проблемы. Для обучения модели машинного обучения обычно требуется несколько этапов (Рисунок

2). К ним относятся предварительная обработка и очистка данных, выбор функций или проектирование, выбор модели, обучение модели на основе обучающих данных и оценка ее производительности на основе данных тестирования или валидации. Производительность модели может быть оценена с использованием различных показателей в зависимости от задачи, таких как точность, безошибочность или среднеквадратичная ошибка.

Рисунок 2 – Этапы обучения модели

1.1.2 Категории алгоритмов

Существуют различные типы алгоритмов машинного обучения, каждый из которых предназначен для решения различных типов задач. Основные категории – это контролируемое обучение, обучение без учителя и обучение с подкреплением.

При контролируемом обучении алгоритм снабжается помеченным набором данных, установленными границами и правилами разработчиком в процессе работы. Каждая точка данных связана с соответствующей целью или результатом. Алгоритм учится сопоставлять входные функции с правильными выходными данными, сводя к минимуму разницу между его прогнозами и фактическими значениями.

Обучение без учителя не предполагает участие разработчика, и как следствие алгоритм не получает маркированных данных, правил и установленных границ.

Цель алгоритма состоит в обнаружении скрытых закономерностей без предварительного знания желаемого результата.

При обучении с подкреплением алгоритм анализирует действия, совершенные машиной, и опирается на полученные результаты при выполнении дальнейших действий путем нахождения корреляций между правильными результатами. Таким образом модель постоянно улучшается, что делает обучение с подкреплением самой продвинутой категорией алгоритмов (“Machine Learning For Absolute Beginners”, Oliver Theobald).

1.1.3 Популярные алгоритмы

1. Линейная регрессия

Линейная регрессия - это модель, относящаяся к контролируемым методам обучения. Она служит ценным инструментом для прогнозирования числовых значений на основе набора входных переменных. Её простота и интерпретируемость делают её популярным выбором для решения многих задач анализа и моделирования данных.

Линейная регрессия направлена на установление линейной связи между зависимой переменной (также известной как целевая переменная) и одной или несколькими независимыми переменными (известными как регрессоры или признаки). Цель состоит в том, чтобы найти оптимально подходящую линию, которая минимизирует расстояние между прогнозируемыми значениями и фактическими значениями в обучающих данных.

Уравнение линии в простой линейной регрессии (Формула 1) может быть выражено так:

$$y = mx + b \quad \text{(Формула 1)}$$

Здесь "y" представляет собой зависимую переменную, "x" обозначает независимую переменную, "m" - наклон линии, а "b" - y-пересечение. Наклон ("m") определяет направление и крутизну линии, а y-пересечение ("b") - точку пересечения линии с осью y.

Для оценки оптимальных значений "m" и "b" в линейной регрессии используется метод наименьших квадратов (OLS). Цель OLS – минимизировать

сумму квадратов разницы между прогнозируемыми и фактическими значениями. Другими словами, он находит линию, которая минимизирует общую ошибку.

Для оценки качества подгонки обычно используются различные метрики. Одним из таких показателей является коэффициент детерминации (R-квадрат), измеряющий долю дисперсии зависимой переменной, которая может быть объяснена независимыми переменными. R-квадрат находится в диапазоне от 0 до 1, причем 1 означает идеальное соответствие.

Линейная регрессия может быть распространена на несколько независимых переменных, что приводит к множественной линейной регрессии. Уравнение множественной линейной регрессии (Формула 2) может быть представлено следующим образом:

$$y = b_0 + b_1x_1 + b_2x_2 + ... + b_nx_n \text{ (Формула 2)}$$

Здесь "b_0" представляет собой y-пересечение, "b_1" - "b_n" - коэффициенты, соответствующие независимым переменным "x_1" - "x_n". Коэффициенты показывают влияние каждой независимой переменной на зависимую переменную при условии, что другие переменные остаются неизменными.

Линейная регрессия применяется в различных областях. Она широко используется в экономике для анализа взаимосвязей между переменными, в финансах для прогнозирования цен на акции, в здравоохранении для прогнозирования результатов лечения пациентов, в маркетинге для прогнозирования продаж.

Хотя линейная регрессия является мощным алгоритмом, у нее есть ограничения. Она предполагает линейную связь между переменными и может не отражать сложные нелинейные закономерности. Кроме того, она может быть чувствительна к высокой корреляции между переменными.

2. Логистическая регрессия

Логистическая регрессия является еще одним контролируемым алгоритмом обучения, но используется для задач бинарной классификации. Она моделирует взаимосвязь между набором независимых переменных и вероятностью наступления

события. Логистическая регрессия ценится за свою простоту, интерпретируемость и эффективность.

В отличие от линейной регрессии, которая прогнозирует непрерывные значения, логистическая регрессия прогнозирует вероятность того, что событие попадет в один из двух классов: 0 или 1.

Модель логистической регрессии применяет логистическую функцию (также известную как сигмоидальная функция) к линейной комбинации входных переменных. Она привязывает любое вещественное число к значению между 0 и 1, что позволяет нам интерпретировать результат как вероятность.

Логистическая функция (Формула 3) определяется так:

$$p = 1 / (1 + e^{(-z)}) \text{ (Формула 3)}$$

Здесь "p" – это вероятность наступления события, а "z" - линейная комбинация входных переменных. Уравнение может быть переписано (Формула 4) как:

$$\ln(p / (1 - p)) = b_0 + b_1x_1 + b_2x_2 + ... + b_nx_n \text{ (Формула 4)}$$

Левая часть уравнения называется logit – это логарифм вероятности наступления события. Правая часть аналогична уравнению линейной регрессии, где 'b_0' представляет собой пересечение, а 'b_1' - 'b_n' обозначают коэффициенты, соответствующие независимым переменным 'x_1' - 'x_n'.

Для оценки оптимальных значений коэффициентов в логистической регрессии обычно используется оценка максимального правдоподобия (MLE) – это метод оценки параметров вероятностного распределения с помощью максимизации функции правдоподобия. Таким образом логистическая регрессия находит коэффициенты, которые соответствуют данным обучения и максимизируют вероятность правильной классификации результатов.

Чтобы сделать прогноз с помощью логистической регрессии, применяется порог принятия решения. Если предсказанная вероятность больше порога, событие классифицируется как 1, в противном случае - как 0.

3. Деревья решений

Деревья решений – это контролируемые алгоритмы, которые используются для классификации и для регрессии. По своей структуре дерево решений похоже на блок-схему, где каждый внутренний узел представляет признак, каждая ветвь - правило принятия решения, а каждый листовой узел - результат или предсказание. Построение дерева предполагает рекурсивное разделение данных на основе значений признаков для максимизации прироста информации или минимизации примесей.

В дереве решений классификации целью является присвоение метки класса каждому элементу. Дерево разделяет данные на основе значений признаков, создавая однородные подмножества с точки зрения меток классов. В регрессионном дереве решений целевая переменная является непрерывной, и цель состоит в том, чтобы предсказать числовое значение. Деревья регрессии обычно используют дисперсию или среднюю квадратичную ошибку (MSE) для оценки качества разбиения и прогнозирования в узлах листа. Дерево рекурсивно разделяет данные на основе значений признаков, чтобы минимизировать дисперсию или MSE в каждом разделе.

Деревья решений способны работать как с категориальными, так и с числовыми признаками, и могут отражать нелинейные взаимосвязи и обрабатывать отсутствующие значения. Они могут также дать представление о важности признаков. Изучая разветвления дерева и связанное с ними уменьшение примесей или дисперсии, можно определить наиболее важные признаки для прогнозирования или классификации.

4. Случайный лес (Random Forest)

Random Forest - это алгоритм, заключающийся в объединении нескольких деревьев решений для составления прогнозов. Он широко используется для задач классификации и регрессии, известен своей высокой точностью, устойчивостью и способностью обрабатывать сложные наборы данных.

Random Forest обучает каждое дерево на случайной выборке подмножества обучающих данных. Кроме того, для каждого дерева при каждом разбиении рассматривается случайное подмножество признаков.

Во время прогнозирования каждое дерево в случайном лесу самостоятельно выдает прогноз, полученный в результате агрегирования ответов множества деревьев. Это улучшает точность прогнозирования в сравнении с одним деревом принятия решений.

Random Forest может обрабатывать данные с большим количеством признаков, а также наборы данных с отсутствующими значениями и позволяет оценить важность признаков на основе того, насколько снижается точность прогнозирования при случайной перестановке выбранного признака.

Для определения оптимального количества деревьев в случайном лесу используются такие методы, как перекрестная валидация или оценка ошибок вне мешка (каждое дерево оценивается на экземплярах, не включенных в его обучающее множество).

Несмотря на все свои сильные стороны, Random Forest может подходить не для всех задач. Его обучение может быть вычислительно дорогим, особенно при большом количестве деревьев и признаков. Также его интерпретируемость может быть ниже, чем у отдельного дерева решений, поскольку проследить связь отдельных прогнозов с конкретными правилами сложнее.

5. Нейронные сети

Нейронные сети – это мощные модели машинного обучения, вдохновленные структурой и функционированием человеческого мозга. Они очень универсальны и могут применяться для решения различных задач, включая классификацию, регрессию, распознавание образов, обработку естественного языка и многое другое.

В основе нейронной сети лежат взаимосвязанные слои искусственных нейронов, известных как узлы. Они получают входные данные, применяют функцию активации для получения выходного сигнала и передают его на следующий слой. Расположение и связь этих слоев определяют архитектуру сети.

Связи между узлами имеют соответствующие веса, которые отражают важность связи. В процессе обучения сеть узнает оптимальные значения этих весов, регулируя их. Этот процесс обычно выполняется с помощью алгоритмов оптимизации, таких как градиентный спуск.

Функция активации, применяемая к каждому узлу, определяет выход этого узла на основе его входов. Обычные функции активации включают сигмоидную функцию, функцию гиперболического тангенса (tanh) и функцию Rectified Linear Unit (ReLU). Выбор функции зависит от задачи и желаемого поведения сети.

Одним из ключевых преимуществ нейронных сетей является их способность к итеративному процессу обучения, благодаря чему они могут извлекать характеристики из исходных данных и делать точные прогнозы или классификации. Эта способность позволяет нейронным сетям хорошо справляться с задачами с большим объемом данных и сложными закономерностями.

Нейронные сети имеют широкий спектр архитектур, включая свёрточные нейронные сети (CNN) для обработки изображений, рекуррентные нейронные сети (RNN) для решения задач, где нечто целостное разбито на части, и трансформеры для задач обработки естественного языка.

6. Алгоритмы кластеризации

Кластеризация – это метод машинного обучения без учителя, суть которого в объединении похожих экземпляров в группы на основе их сходства. В отличие от контролируемого обучения, кластеризация не требует маркированных данных и нацелена на обнаружение закономерностей в самих данных.

Цель кластеризации - разделить набор данных на группы или кластеры, где экземпляры в одном кластере более похожи друг на друга, чем экземпляры в других кластерах. Алгоритмы кластеризации стремятся максимизировать внутрикластерное сходство и минимизировать межкластерное сходство.

Алгоритмы кластеризации можно разделить на два основных типа: иерархическая кластеризация и плоская. Иерархическая кластеризация строит систему вложенных разбиений. В результате получается дерево кластеров, показывающее связи между кластерами на разных уровнях детализации. Плоские алгоритмы строят одно разбиение объектов на кластеры.

Различные внутренние и внешние метрики оценки могут дать представление о качестве кластеров. Внутренние метрики оценивают компактность и разделение кластеров, а внешние метрики сравнивают результаты кластеризации с известными метками классов, если таковые имеются.

Кластеризация имеет широкий спектр применения в различных областях. Она может использоваться для сегментации изображений, кластеризации документов, обнаружения аномалий и т. д. Группируя похожие экземпляры вместе, кластеризация

помогает обнаружить значимые закономерности, выявить выбросы и получить представление об основной структуре данных.

7. Q-обучение

Q-обучение – это один из алгоритмов категории методов обучения с подкреплением. Это свободный от моделей алгоритм, изучающий оптимальную функцию "действие-значение", также известную как Q-функция, которая оценивает ожидаемое вознаграждение за принятие определенного действия. Обучение состоит из двух компонентов – агента и окружения. Q-функция обычно представлена в виде таблицы, значения Q обновляются итеративно на основе опыта агента (Рисунок 3).

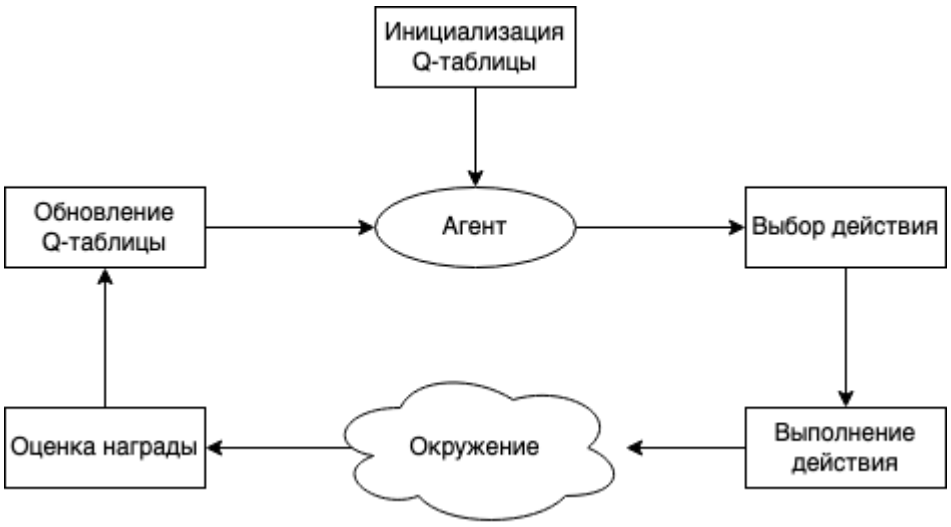

Рисунок 3 – Реализация Q-обучения

В процессе обучения агент исследует окружающую среду, предпринимая какие-то действия. Первоначально агент может использовать высокую скорость исследования для обнаружения различных состояний и действий. По мере обучения скорость исследования обычно снижается, и агент больше фокусируется на использовании полученных знаний для получения максимального вознаграждения.

Q-обучение имеет несколько преимуществ. Так как это алгоритм без модели, то он не требует предварительных знаний о среде. Он может работать со средами с большими или непрерывными пространствами состояний и действий, используя методы аппроксимации функций.

Однако Q-обучение имеет недостатки. Оно может быть чувствительным к начальным условиям и может потребовать большого количества взаимодействий с

окружающей средой, чтобы прийти к оптимальной политике. Компромисс между исследованием и эксплуатацией может иметь решающее значение для баланса между исследованием новых состояний и эксплуатацией полученных знаний.

Расширения и вариации Q-обучения были разработаны для решения конкретных задач и повышения эффективности. Некоторые яркие примеры включают глубокие Q-сети (DQN), которые объединяют Q-обучение с глубокими нейронными сетями для работы с высокоразмерными пространствами состояний, и двойное Q-обучение, которое смягчает переоценку Q-значений.

1.2 Анализ существующих музыкальных плееров

Разные методы машиного обучения широко используются в различных областях. Для выявления функций, реализованных в существующих музыкальных проигрывателях посредством применения машинного обучения, был проведен анализ программ-аналогов (Таблица 1).

Таблица 1 – анализ существующих музыкальных проигрывателей

№	Название программы-аналога	Основной функционал	Наличие необычных функций	Используется ли машинное обучение?
1	Dopamine	Есть возможность воспроизведения файлов различны форматов, можно выбрать тему оформления. Можно создавать свои плейлисты.	Есть демонстрация облака популярности музыки (чем больше прослушиваний у песни – тем больше ее обложка в облаке). Песни автоматически сортируются по исполнителям, жанрам и альбомам, если им присвоены	Нет

			теги.	
2	Winamp	Можно проигрывать аудио-файлы различных форматов, также можно воспроизводить видео. Имеется большое количество настроек звука. Можно кастомизировать интерфейс.	Сортировка песен по тегам (также есть возможность редактировать теги). Имеется модуль визуализации.	Нет
3	MusicBee	Воспроизведение популярных аудио-форматов, поддержка импорта музыки из библиотеки WM, наличие эквалайзера, создание плейлистов.	Сортировка по тегам, которые можно автоматически присвоить при наличии доступа к сети. Автоматическое создание плейлистов на основе часто прослушиваемой музыки или по заданным параметрам.	Нет
4	FooBar2000	Поддержка большого количества аудиоформатов, работа с фонотекой, создание плейлистов, настраиваемые горячие клавиши, возможность прямого вывода на	Сортировка по тегам	Нет

		звуковую карту		
5	AIMP	Одновременная работа с несколькими плейлистами, поддержка 44 аудиоформатов, наличие эквалайзера, встроенные звуковые эффекты, встроенный редактор тегов, настройка горячих клавиш, воспроизведение интернет-радио	Автоматическая нормализация уровня громкости, сортировка песен, караоке-плагин	Нет
6	MediaMonkey	Поддержка большого количества аудиоформатов, создание плейлистов, наличие эквалайзера, конвертация файлов, история воспроизведения, возможность кастомизации интерфейса, поддержка подкастов	Автоматическое воспроизведение музыки по заданным критериям, создание отчётов по музыкальной коллекции, сортировка песен по тегам	Нет

1.3 Используемые в существующих решениях алгоритмы машинного обучения

При анализе существующих музыкальных проигрывателей для ОС Windows было выявлено, что ни один из них не использует в работе методы машинного обучения. Некоторые плееры предлагают возможности сортировки композиций по

жанрам, датам добавления и исполнителям. Такие классификации в основном основаны на предварительно присвоенных тегах, что ограничивает их точность и гибкость. Они не используют методы машинного обучения для более сложных классификаций, анализа музыкальных данных или разработки других функций проигрывателя.

Следовательно, ни один из рассмотренных проигрывателей не предоставляет возможность генерации индивидуальных обложек для альбомов или функцию распознавания текста песни. Это ограничение снижает визуальную привлекательность и возможности пользователей воспринимать музыку в более интерактивном формате.

Однако стоит отметить, что методы машинного обучения широко применяются в других областях музыкальной индустрии, в частности, в стриминговых сервисах, где они используются для генерации плейлистов, рекомендаций и предсказания музыкальных предпочтений пользователей. Например, используется:

1. коллаборативная фильтрация: этот подход использует историю прослушивания пользователей для определения их музыкальных предпочтений и создания рекомендаций на основе схожести между пользователями. Он анализирует данные о том, какую музыку слушают различные пользователи, и на основе этой информации предлагает рекомендации, основываясь на предпочтениях других пользователей с похожими вкусами.
2. Содержательная фильтрация: этот метод использует характеристики музыкальных треков, такие как жанр, инструменты, темп и т.д., для создания рекомендаций на основе схожести между треками. Он анализирует музыкальные особенности и характеристики каждого трека и предлагает рекомендации на основе их сходства с уже прослушанными.
3. Нейронные сети: эти метод позволяет анализировать более сложные и абстрактные характеристики музыки, используя глубокие нейронные сети. Например, нейронные сети могут быть обучены распознавать эмоциональный характер музыки или предсказывать, какие треки будут нравиться пользователю, исходя из его предыдущих предпочтений.
4. Анализ текста и обработка естественного языка: эти методы могут использоваться для анализа текстовых данных, таких как название и текст песни, имя исполнителя. Алгоритмы обработки естественного языка могут распознавать смысловые аспекты текста и использовать их для рекомендации или категоризации музыкальных треков.

Это лишь некоторые из методов машинного обучения, применяемых в стриминговых сервисах. Их применение в оффлайн музыкальных проигрывателях может открыть новые возможности для пользователей, позволяя им персонализировать опыт прослушивания музыки.

1.4 Выводы к главе 1

Проведенный анализ предметной области и существующих музыкальных проигрывателей позволяет сделать ряд выводов:

1. Машинное обучение как никогда актуально, используется в различных сферах и содержит множество методов.
2. Использование методов машинного обучения является трендом среди музыкальных стриминговых сервисов, однако в оффлайн музыкальных проигрывателях эти методы не применяются совсем.
3. Использование методов машинного обучения в оффлайн проигрывателях необходимо для улучшения пользовательского опыта их использования.

Из перечисленных выводов вытекает необходимость создания умного музыкального плеера, в котором применяются методы машинного обучения. Вторая глава дипломной работы посвящена выработке требований к нему, разработке прототипа и демонстрации результатов.

Глава 2. Разработка умного музыкального плеера

2.1 Требования к продукту

В процессе анализа предметной области и существующих музыкальных плеер было составлено техническое задание, содержащее в себе все требования к умному музыкальному плееру.

Две основные функции, отличающие разрабатываемый продукт от других, представленных на рынке, – это генерация изображений и распознавание текста композиций.

Ниже представлены выжимки из технического задания. Полный его текст представлен в приложении 1.

Назначение: программный продукт «Умный музыкальный плеер» предназначен для воспроизведения музыкальных композиций.

Цель создания программного продукта: разработать программный продукт, позволяющий повысить интерактивность и персонализированность прослушивания музыки в оффлайн-режиме.

Целевая аудитория: пользователи от 6 до 60 лет

Основные задачи:

1. Воспроизведение музыкальных композиций в оффлайн-режиме
2. Генерация обложек к музыкальным композициям при их отсутствии или по желанию пользователя
3. Распознавание текста композиции

Требования к стилистическому оформлению программного продукта: стилистическое оформление должно соответствовать основному назначению программного продукта. Дизайн, структура программного продукта должны

вызывать у пользователя только положительные эмоции. Это обеспечивается благодаря использованию определенной цветовой гаммы и эргономичности построения графических элементов интерфейса.

Требования к дизайну: дизайн должен быть индивидуальным и запоминающимся.

Обязательные графические элементы программного продукта:

1. основная панель управления, содержащая кнопки “воспроизведение”, “вперед”, “назад”
2. ползунок для регулировки громкости
3. шкала, отображающая прогресс воспроизведения композиции
4. кнопка для отображения текста песни
5. холст для отображения обложки к композиции размером 256x256

2.2 Системные требования

Для локальной генерации изображений и распознавания речи требуется хорошее аппаратное обеспечение. Это связано с вычислительными потребностями, возникающими при обучении моделей. Мощный компьютер позволяет эффективно выполнять работу, тем самым способствуя получению высококачественных результатов и сокращению времени обработки.

Так как проигрыватель должен работать локально, то минимальные системные требования к машине пользователя следующие:

1. ОС: Windows 10/11 или MacOS (на устройствах с чипсетом M1/M2)
2. Оперативная память: 8 Гбайт
3. Видеокарта: Nvidia 10xx или новее, оснащенная не менее 4 Гбайт видеопамяти
4. Свободное дисковое пространство: 20 Гбайт

2.3 Интерфейс приложения

Для удобства использования интерфейс не должен быть перегружен и обязан быть интуитивно понятным и простым. Для этого было принято решение остановиться на цветовой гамме из 2-3 контрастных цветов, текст на кнопках по возможности заменить на символы.

В результате был спроектирован интерфейс, соответствующих требованиям (Рисунок 4).

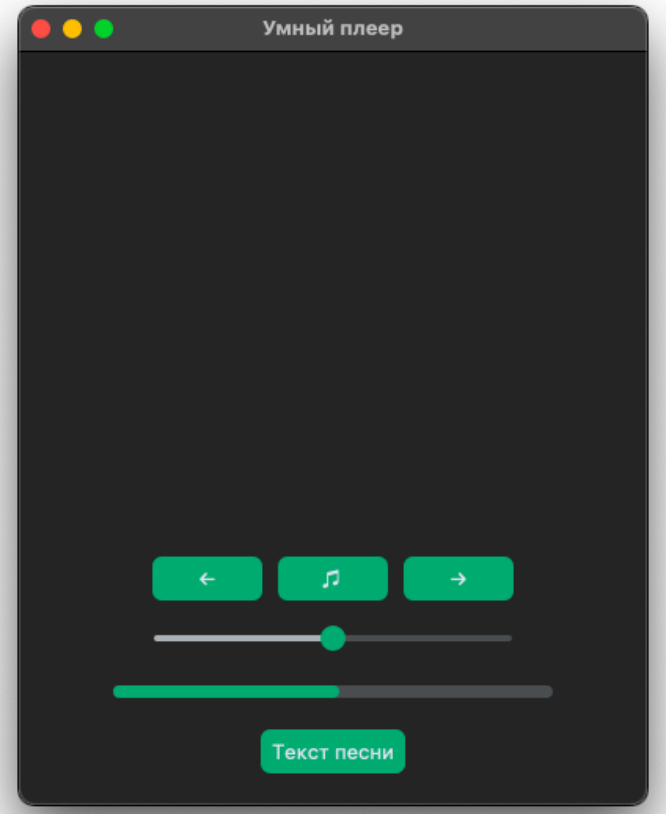

Рисунок 4 – Интерфейс музыкального плеера

2.4 Описание основных этапов работы пользователя с системой

Основные этапы работы пользователя с приложением такие:

1. Пользователь устанавливает приложение
2. Пользователь загружает музыкальные композиции в определенную директорию
3. Пользователь запускает приложение
4. Пользователь может проигрывать музыку из определенной директории
5. При необходимости пользователь может переключать композиции, просматривать текст песни

На основе этих этапов был также составлен use-case сценарий (см. Рисунок 5):

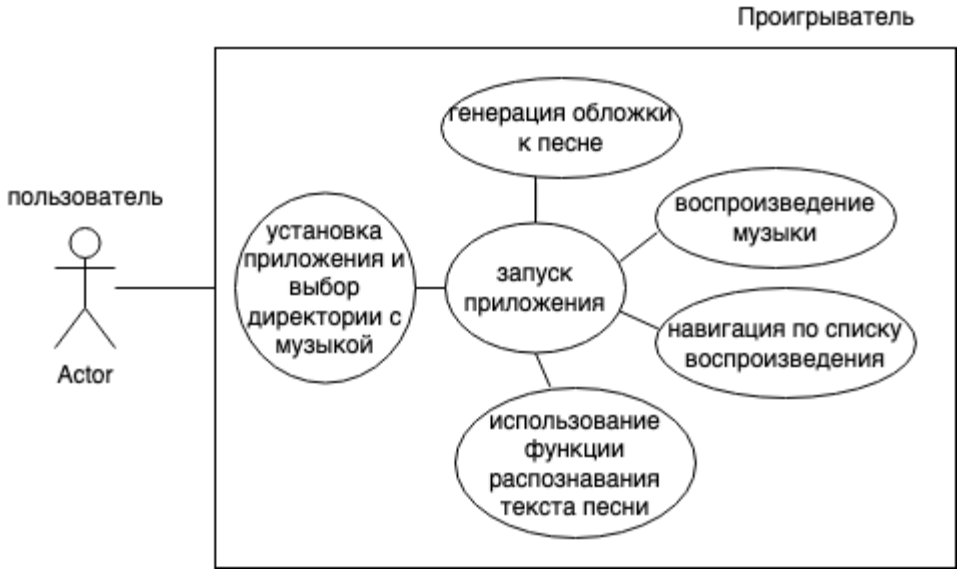

Рисунок 5 – Use-case сценарий

2.5 Методология управления разработкой

Методология разработки программного обеспечения - это четко определенный подход, используемый для руководства процессом создания программного продукта, который помогает организовать и структурировать работу над проектом. Можно назвать это системой из лучших практик, стратегий и инструментов, которые используются для обеспечения эффективного управления требованиями проекта, точного отслеживания хода работы, менеджмента задач и времени.

Выбор подходящей методологии разработки программного обеспечения зависит от нескольких факторов, включая масштаб и сложность проекта, доступные сроки, имеющиеся ресурсы и необходимый уровень гибкости. Например, в методологии Waterfall особое внимание уделяется тщательному планированию и всеобъемлющему документированию. Напротив, Agile-методологии, такие как Scrum, Kanban и Lean, делают упор на адаптивность, обеспечивая большую гибкость на протяжении всей разработки.

Для работы над приложением “Умный музыкальный плеер” была выбрана методология Kanban. Она широко используется в управлении проектами Agile, обеспечивает гибкий подход к управлению проектом, позволяет удобно отслеживать прогресс, оптимизировать рабочий процесс.

Для управления задачами в этой методологии используются Kanban доски, они состоят из колонок, представляющих различные этапы работы, такие как "Выполнить", "В работе" и "Выполнено". Задачи представлены карточками, они перемещаются по этим колонкам по мере выполнения работы. Такое визуальное представление дает четкое и интуитивно понятное представление о текущих задачах команды и их статусе.

Одним из преимуществ Kanban является его гибкость. Она позволяет быстро адаптироваться к изменяющимся приоритетам и требованиям, что особенно ценно для небольших команд, работающих над новым проектом, в который в процессе могут вноситься изменения.

2.6 Разработка прототипа приложения

2.6.1 Используемые инструменты

Для разработки прототипа приложения использовался язык программирования Python. “... Это интерпретируемый, интерактивный, объектно-ориентированный язык программирования. Он включает в себя модули, исключения, динамическую типизацию, динамические типы данных очень высокого уровня и классы. Он поддерживает множество парадигм программирования, помимо объектно-ориентированного программирования, таких как процедурное и функциональное программирование. Python сочетает в себе удивительную мощь и очень понятный синтаксис.” (Документация)

Python был выбран в качестве языка программирования для разработки данного проекта по множеству причин. Прежде всего, на решение повлияла читабельность и простота языка. Этот фактор удобочитаемости значительно сокращает время и усилия, необходимые для разработки.

Кроссплатформенность языка также послужила одним из факторов выбора в его пользу. Поддерживая основные ОС, такие как Windows и Unix-подобные системы, Python обеспечивает бесперебойную работу разработанного ПО в различных средах, что устраняет проблемы совместимости и позволяет охватывать широкую базу пользователей.

Помимо перечисленных выше преимуществ, популярность Python в области машинного обучения (ML) стала решающим фактором при выборе. Язык предоставляет широкий спектр библиотек и фреймворков, разработанных конкретно для решения задач ML.

2.6.2 IDE

В качестве интегрированной среды разработки (IDE) для разработки прототипа был выбран PyCharm. Эта среда обладает чистым и интуитивно понятным пользовательским интерфейсом, что в сочетании с высокой производительностью обеспечивает продуктивную работу над кодом.

Одним из основных преимуществ среды является её глубокая интеграция с языком Python. Будучи специализированной для него IDE, PyCharm предоставляет обширный набор инструментов и возможностей, предназначенных специально для разработки на нём. Они включают в себя, например, подсветку синтаксиса и анализ кода с возможным последующим рефакторингом, что позволяет писать чистый и читабельный код.

Также IDE предоставляет расширенные инструменты отладки, такие как точки останова, проверка переменных и пошаговое выполнение, которые облегчают выявление и решение проблем в коде, что позволяет повысить общее качество и надежность разрабатываемого ПО.

2.6.3 Библиотеки и фреймворки

Помимо стандартных встроенных библиотек, таких как io, math, os, time и др., в разработке прототипа умного музыкального плеера были использованы сторонние решения, установленные вручную.

Одна из таких библиотек – customtkinter. Это расширение для стандартной библиотеки tkinter, встроенной в Python, которое позволяет разрабатывать визуально приятный современный интерфейс. Одной из особенностей customtkinter является то, что она позволяет разрабатываемому приложению подстраивать цвет интерфейса под тему, выбранную пользователем в качестве основной системной темы. Так, если пользователь выставил тёмную тему в Windows – тема приложения подстроится и тоже будет тёмной.

Для реализации функционала проигрывателя (воспроизведение музыки в частности) была использована библиотека pygame, предназначенная для разработки игр и мультимедийных приложений.

Для работы с изображениями (обложками музыкальных композиций) использовалась библиотека PIL (Python Imaging Library).

Так как для локальной генерации изображений и распознавания речи нужны большие вычислительные мощности, для разработки прототипа проигрывателя было принято решения использовать сторонние инструменты от Google и OpenAI для решения этих задач. Таким образом будет получен код, для которого при разработке полноценного умного музыкального плеера нужно будет реализовать эти две функции и интегрировать их вместо имеющихся в прототипе.

Для генерации обложек к песням было использовано API инструмента DALL-E от OpenAI, как было упомянуто ранее. Для распознавания текста песен было использование API инструмента Google Speech-to-Text. Принципы их работы будут рассмотрены подробно в следующем пункте работы.

2.6.4 Код

Первое, что было сделано – это разработка интерфейса средствами tkinter и customtkinter. Сначала было создано основное окно и установлены настройки customtkinter:

```
customtkinter.set_appearance_mode("System")  # Modes: system (default), light, dark
customtkinter.set_default_color_theme("green")       #   Themes:   blue (default), dark-blue, green

root = customtkinter.CTk()  # создаем окно
root.title('Умный плеер')  # даем название окну
root.geometry('400x480')  # устанавливаем размер окна
```

Затем в этом окне были размещены необходимые элементы графического интерфейса:

```
back_button     =     customtkinter.CTkButton(master=root,     text='←', command=skip_back, width=70)
back_button.place(relx=0.3, rely=0.7, anchor=tkinter.CENTER)

play_button     =     customtkinter.CTkButton(master=root,     text='♫', command=play_music, width=70)
```

```
play_button.place(relx=0.5, rely=0.7, anchor=tkinter.CENTER)

forward_button = customtkinter.CTkButton(master=root, text='→', command=skip_forward, width=70)
forward_button.place(relx=0.7, rely=0.7, anchor=tkinter.CENTER)

volume_slider = customtkinter.CTkSlider(master=root, from_=0, to=1, command=volume, width=240)
volume_slider.place(relx=0.5, rely=0.78, anchor=tkinter.CENTER)

progress_bar = customtkinter.CTkProgressBar(master=root, width=280)
progress_bar.place(relx=0.5, rely=0.85, anchor=tkinter.CENTER)

lyrics_button = customtkinter.CTkButton(master=root, text='Текст песни', command=show_lyrics, width=70)
lyrics_button.place(relx=0.5, rely=0.93, anchor=tkinter.CENTER)
```

Далее была реализована функция воспроизведения музыки:

```
def play_music():
    threading()     # Использование многопоточности для возможности одновременно проигрывать музыку и отслеживать
    # прогресс воспроизведения
    global n  # Индекс песни
    if n > len(list_of_songs):
        n = 0
    song_name = list_of_songs[n]  # Получение имени песни из списка песен по индексу "n"
    get_cover(song_name)  # Вызов функции get_cover для получения обложки песни

    stripped_string = song_name[6:-4]
    song_name_label = tkinter.Label(text=stripped_string, bg="#222222", fg='white') # Создание метки для отображения
    # имени песни
    song_name_label.place(relx=.5, rely=.62, anchor=tkinter.CENTER)

    pygame.mixer.music.load(str(song_name))
    pygame.mixer.music.play(loops=0)  # Воспроизведение песни без повторения
    pygame.mixer.music.set_volume(.5  # Установка громкости воспроизведения
```

Были также реализованы простые функции навигации по списку воспроизведения skip_back() и skip_forward() (переключиться на одну песню назад или вперёд соответственно):

```
def skip_back():
    global n
    n -= 1
    play_music()


def skip_forward():
    global n
    n += 1
    play_music()
```

рогресса воспроизведения песни (progress()):

Помимо этого были реализованы функции возможности регулировки громкости (volume()) и отслеживания прогресса воспроизведения (progress()):

```
def volume(value):
    pygame.mixer.music.set_volume(value)

def progress():
    a = pygame.mixer.Sound(f'{list_of_songs[n]}')
    song_len = a.get_length() * 3
    for i in range(0, math.ceil(song_len)):
        time.sleep(.3)
        progress_bar.set(pygame.mixer.music.get_pos() / 1000000)
```

Функция получения обложки песни, которая вызывается в функции воспроизведения музыки, была реализована с помощью DALL-E от OpenAI таким образом:

```
def get_cover(song_name):
    load_dotenv()
    openai.api_key = os.environ.get('OPENAI_KEY')
```

```
    print(song_name[6:-4].split("- ", 1)[1])

    # Генерируем обложку по текстовому запросу
    # "/название песни/ + в стиле обложки музыкального альбома"
    response = openai.Image.create(
        prompt=song_name[6:-4].split("- ", 1)[1] + ' in style of music
album cover',
        n=1,
        size="256x256"
    )

    image_url = response['data'][0]['url']  # Получаем адрес изображения
    response = requests.get(image_url)  # Получаем само изображение

    # Размещаем изображение в главном окне
    image = Image.open(io.BytesIO(response.content))
    load = ImageTk.PhotoImage(image)
    label1 = tkinter.Label(root, image=load)
    label1.image = load
    label1.place(relx=.19, rely=.06)
```

DALL-E состоит из нейронных сетей GPT (Generative Pre-trained Transformer), главная задача которой – обработка естественного языка, и VQ-GAN, способная преобразовывать изображение в сетку векторов (токенов) и наоборот (Рисунок 6).

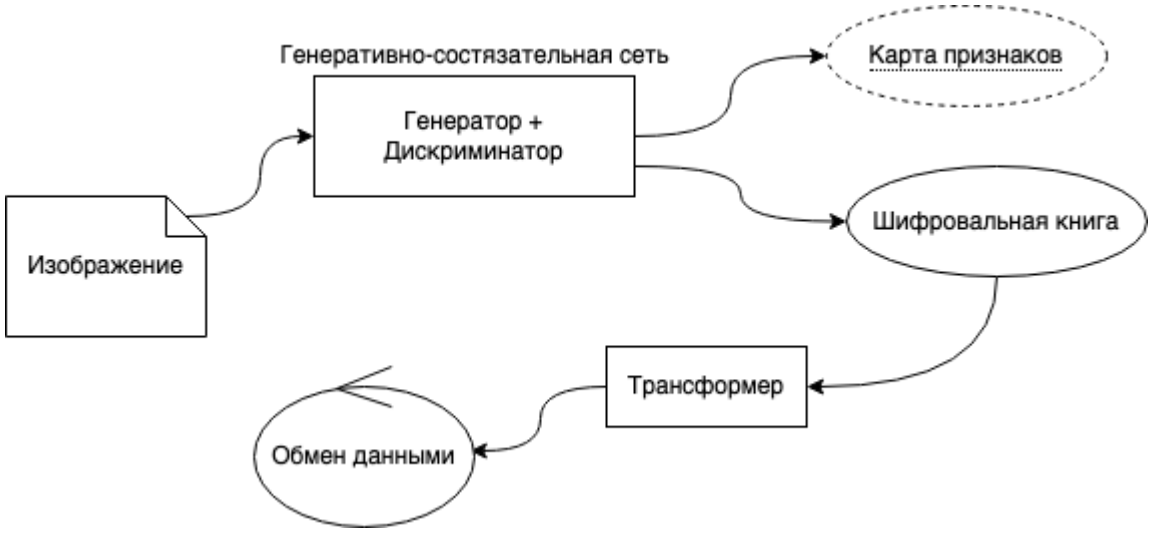

Рисунок 6 – Процесс работы VQ-GAN

GPT привязывает к каждому из них набор векторов (Key и Query – значимость в последовательности при рассмотрении извне/из этого токена соответственно и Value – репрезентация токена). Векторы Key и Query умножаются друг на друга для получения коэффициентов значимости каждого вектора, которые затем умножаются на Value для получения суммы Внимания последовательности (Внимание является аналогом когнитивного внимания и позволяет находить связи между токенами, а также предсказывать продолжение этих токенов).

Таким образом моделируется последовательность токенов, которую при обучении GPT на небольшом датасете можно преобразовать в токены, подобные VQ-GAN. Затем декодер VQ-GAN преобразовывает их в изображение с соответствующими токенами.

Для песни с названием “Strawberry fields forever” (“Клубничные поля навсегда”) в результате интеграции DALL-E в разрабатываемый прототип плеера были получены такие результаты (Рисунок 7, 8):

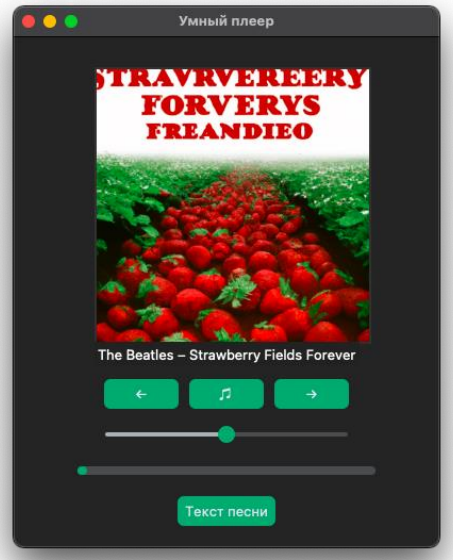

Рисунок 7 – Пример генерации обложки №1

Рисунок 8 – Пример генерации обложки №2

Для песни с названием “” (“Маленькая черная субмарина”) были получены такие результаты (Рисунок 9, 10):

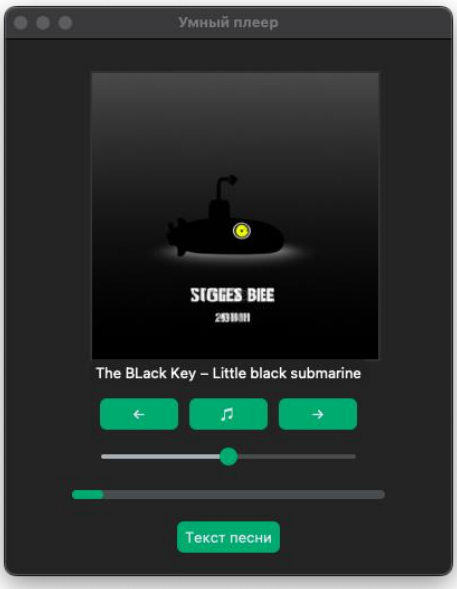

Рисунок 9 – Пример генерации обложки №3

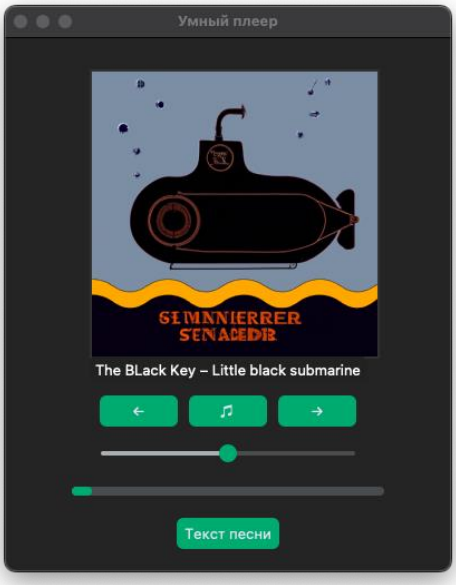

Рисунок 10 – Пример генерации обложки №4

Функция распознавания текста песен была реализована таким образом: сначала была разработана функция lyrics_recognition(). Текст распознается при помощи инструмента Google Speech Recognition, использующего технологию Speech-to-Text (STT). Она работает на основе нейронной сети, обрабатывающей речь и создающей соответствующий текст. Нейронная сеть обучается на наборе данных записей человеческой речи в паре с аннотированным текстом, что позволяет ей ассоциировать акустические паттерны с определенными буквами и словами, изображенными в виде спектрограммы. На результаты распознавания в основном влияет объём обучающих данных и качество входного сигнала. Для повышения точности транскрипции используются языковые модели, учитывающие контекст текста.

Функция выглядит так:

```
def lyrics_recognition(song_name):

    client = speech.SpeechClient()

    gcs_uri = 'gs://my_player_bucket/' + song_name[6::]
    config = speech.RecognitionConfig(
        encoding=speech.RecognitionConfig.AudioEncoding.LINEAR16,
        enable_automatic_punctuation=True,
        audio_channel_count=2,
```

```
        language_code='en-US',
    )
    audio = speech.RecognitionAudio(uri=gcs_uri)
    operation = client.long_running_recognize(config=config,
audio=audio)

    response = operation.result(timeout=90)
    return response.results
```

Также было создано всплывающее окно в функции show_lyrics(), которая привязана к кнопке в главном окне. В результате слова распознаются не идеально, однако суть песни обычно улавливается. Ниже представлен пример распознавания речи в песне (Рисунок 11).

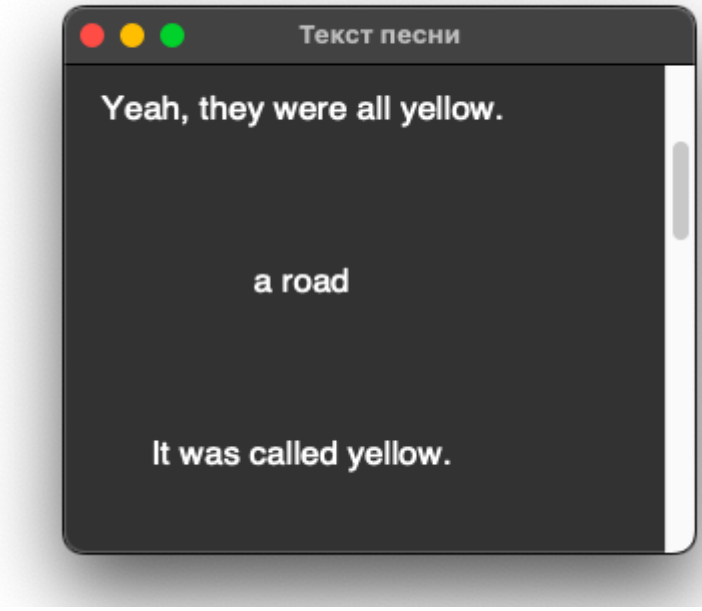

Рисунок 11– Пример распознавания текста песни.

2.7 Выводы к главе 2

При выработке требований к умному музыкальному плееру и создании прототипа было выполнено следующее:

1. Выработаны требования к продукту, включая техническое задание, макет графического интерфейса, use-case сценарий.
2. Сформулированы системные требования для машины, на которой будет производиться разработка умного музыкального плеера.
3. Разработан прототип приложений “Умный музыкальный плеер”, обладающее необходимым базовым функционалом.

Заключение

При выполнении данной дипломной работы был разработан прототип программного продукта «Умный музыкальный плеер», который способен воспроизводить музыкальные композиции, генерировать к ним уникальные обложки и распознавать тексты песен.

В процессе выполнения работы были решены следующие задачи:

1. Был проведен анализ предметной области и существующих программ-аналогов.
2. На основе проведенного анализа были разработаны технические требования к умному музыкальному плееру и его интерфейсу.
3. Был разработан прототип умного музыкального плеера.

Таким образом, цель дипломной работы полностью реализована.

Одной из важнейших перспектив развития проекта является разработка функций генерации изображений и распознавания речи и внедрения их в прототип вместо имеющихся функций, основанных на сторонних решениях.

Кроме того, может быть реализована разработка новых функций, например: воспроизведение видео, автоматическая генерация плейлистов на основе различных характеристик композиций, возможность перевода распознанных иностранных текстов песен, функция настройки звука (эквалайзер).

Литература

1. Как работает DALL-E // habr.com: сайт. – URL: https://habr.com/ru/companies/ruvds/articles/687508/ (дата обращения: 23.04.2023)
2. Копырин, А. С. Программирование на Python : учебное пособие / А. С. Копырин, Т. Л. Салова. — Москва : ФЛИНТА, 2021. — 48 с. — ISBN 978-5-9765-4753-7. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/182960 (дата обращения: 05.03.2023). — Режим доступа: для авториз. пользователей.
3. Лекун, Я. Как учится машина: Революция в области нейронных сетей и глубокого обучения / Я. Лекун. — Москва : Альпина Паблишер, 2021. — 351 с. — ISBN 978-5-907470-52-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/213980 (дата обращения: 06.04.2023)
4. Машинное обучение / Б. Хенрик, Д. Уиттон, Т. Хасти, Р. Тибширани. – Санкт-Петербург : Питер, 2017. – 336 с. – ISBN 978-5-496-02989-6
5. Омельяненко, Я. Эволюционные нейросети на языке Python : руководство / Я. Омельяненко ; перевод с английского В. С. Яценкова. — Москва : ДМК Пресс, 2020. — 310 с. — ISBN 978-5-97060-854-8. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/179494 (дата обращения: 06.04.2023). — Режим доступа: для авториз. пользователей.
6. Плас Дж. В. Python для сложных задач: наука о данных и машинное обучение. / Дж. В. Плас. – Санкт-Петербург : Питер, 2018. – 576 с.
7. Полупанов, Д. В. Программирование в Python 3 : учебное пособие / Д. В. Полупанов, С. Р. Абдюшева, А. М. Ефимов. — Уфа : БашГУ, 2020. — 164 с. — ISBN 978-5-7477-5230-6. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/179915 (дата обращения: 05.02.2023). — Режим доступа: для авториз. пользователей.
8. Рашка, С. Python и машинное обучение: крайне необходимое пособие по новейшей предсказательной аналитике, обязательное для более глубокого понимания методологии машинного обучения : руководство / С. Рашка ;

перевод с английского А. В. Логунова. — Москва : ДМК Пресс, 2017. — 418 с. — ISBN 978-5-97060-409-0. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/100905 (дата обращения: 05.02.2023). — Режим доступа: для авториз. пользователей.

9. Северанс, Ч. Р. Python для всех / Ч. Р. Северанс ; перевод с английского А. В. Снастина. — Москва : ДМК Пресс, 2022. — 262 с. — ISBN 978-5-93700-104-7. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/241115 (дата обращения: 05.02.2023). — Режим доступа: для авториз. пользователей.
10. Уилкс, М. Профессиональная разработка на Python / М. Уилкс ; перевод с английского А. А. Слинкина. — Москва : ДМК Пресс, 2021. — 502 с. — ISBN 978-5-97060-930-9. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/241121 (дата обращения: 05.02.2023). — Режим доступа: для авториз. пользователей.
11. Уткин Л. В. Машинное обучение (Machine Learning) / Л. В. Уткин. – Санкт-Петербург : Санкт-Петербургский политехнический университет Петра Великого, 2018. – 43 с.
12. Чару, А. Нейронные сети и глубокое обучение: учебный курс. / А. Чару,. – Санкт-Петербург : Диалектика, 2020. – 752 с.
13. «Яндекс Музыка» представила бесконечную мелодию для концентрации под названием «Нейромузыка» — Текст : электронный // habr.com : [сайт]. — URL: https://habr.com/ru/news/t/680394/ (дата обращения: 08.03.2023).
14. Bonaccorso, G. Machine Learning Algorithms / G. Bonaccorso. – Livery Place 35 Livery Street Birmingham B3 2PB, UK : Packt Publishing Ltd., 2017. – 155-167 с. – ISBN 978-1-78588-962-2.
15. Briot, Jean-Pierre Deep Learning Techniques for Music Generation -- A Survey / Jean-Pierre Briot. — Текст : электронный // arxiv.org : [сайт]. — URL: https://arxiv.org/abs/1709.01620 (дата обращения: 05.02.2023).
16. DALL·E 2 // OpenAI : сайт. – URL: https://openai.com/product/dall-e-2 (дата обращения: 10.05.2023)
17. Machine Learning / T. Michel, B. M. Бухштабер, И. С. Енюков, Л. Д. Мешалкин. – New York : McGraw Hill, 2006. – 250 с.

18. Mueller, J. P. Machine Learning For Dummies / J. P. Mueller, L. Massaron. – 2nd edition. – 111 River Street, Hoboken, NJ : John Wiley & Sons, Inc, 2021. – 467 с. – ISBN 978-1-119-72401-8.
19. Python Software Foundation. Python programming language official website [сайт]. Python.org; [Дата обращения: 05.04.2023]. URL: https://python.org
20. Shalev-Shwartz, S. Understanding of Machine learning. From Theory to Algorythms / S. Shalev-Shwartz, S. Ben-David. – New York : Cambrige University Press, 2014. – 446 с. – ISBN 978-1-107-05713-5.
21. Theobald, O. Machine Learning For Absolute Beginners / O. Theobald. – Japan : Independently published, 2017. – 128 с.
22. Understanding VQ-VAE (DALL-E Explained Pt. 1) : сайт. – URL: https://ml.berkeley.edu/blog/posts/vq-vae/ (дата обращения: 11.05.2023)
23. Yu, D Automatic Speech Recognition / D Yu, L Deng. – Springer-Verlag London : Springer London, 2014. – 321 с. – ISBN 978-1-4471-5779-3.

Приложения

ПРИЛОЖЕНИЕ 1

Техническое задание

на разработку умного музыкального плеера

Исполнитель: студентка 4 курса ИИТиТО Бражкина Анастасия Дмитриевна

Заказчик:

Санкт-Петербург
2023

1. Общие сведения

1.1. Полное наименование программного продукта
Полное наименование программного продукта – «Умный музыкальный плеер».

2. Этапы и сроки создания

2.1. Общий срок работ по созданию программного продукта составляет 4 календарных месяца (25.01 – 25.05)

· Создание графического дизайна приложения – 2 календарных дня
· Реализация графического интерфейса с помощью средств языка Python – 5 календарных дней
· Реализация основных функций проигрывателя (воспроизведение музыкальных композиций, навигация по списку воспроизведения, регулировка громкости звука) – 7 календарных дней
· Реализация функции генерации обложек к композициям – 1,5 календарных месяца
· Реализация функции распознавания текста музыкальной композиции — 1,5 календарных месяца
· Тестирование и отладка приложения – 14 календарных дней

3. Назначение и цели создания программного продукта

3.1. Назначение

Программный продукт «Умный музыкальный плеер» предназначен для воспроизведения музыкальных композиций.

3.2. Цель создания программного продукта

Разработать программный продукт, позволяющий повысить интерактивность и персонализированность прослушивания музыки в оффлайн-режиме.

3.3. Целевая аудитория

Целевая аудитория программного продукта представлена следующими группами пользователей:

· пользователи от 6 до 60 лет

3.4. Основные задачи

Разработанный программный продукт должен обеспечивать реализацию следующих

задач:

· Воспроизведение музыкальных комопозиций в оффлайн-режиме
· Генерация обложек к музыкальным композициям при их отсутствии или по желанию пользователя
· Распознавание текста композиции

4. Технологические требования

4.1. Требования к стилистическому оформлению программного продукта

Стилистическое оформление должно соответствовать основному назначению программного продукта. Дизайн, структура программного продукта должны вызывать у пользователя только положительные эмоции. Это обеспечивается благодаря использованию определенной цветовой гаммы и эргономичности построения графических элементов интерфейса.

4.2. Требования к дизайну

Дизайн должен быть индивидуальным и запоминающимся.

Интуитивно-понятная система навигации является основным условием при разработке программного продукта. Ее использование должно обеспечить легкий и быстрый поиск нужных кнопок, свободное ориентирование по интерфейсу.

4.3.Обязательные графические элементы программного продукта

· основная панель управления, содержащая кнопки “воспроизведение”, “вперед”, “назад”
· ползунок для регулировки громкости
· шкала, отображающая прогресс воспроизведения композиции
· кнопка для отображения текста песни
· холст для отображения обложки к композиции размером 256x256
4.4. Программные и аппаратные требования

ОС: Windows 10/11 или MacOS (на устройствах с чипсетом M1/M2)
Оперативная память: не менее 8 Гбайт
Видеокарта: Nvidia 10xx или новее, оснащенная не менее 4 Гбайт видеопамяти
Свободное дисковое пространство: 20 Гбайт

5. Структура и описание компонентов

Программный продукт состоит из следующих компонентов:

1. Основное окно, в котором осуществляется воспроизведение музыки, навигация по списку воспроизведения, регулировка громкости, кнопка для отображения текста песни
2. Окно с текстом песни, которое отображается при нажатии на определенную кнопку и закрывается пользователем при необходимости.