МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра информатики и программирования

СОЗДАНИЕ РЕКОМЕНДАТЕЛЬНОЙ СИСТЕМЫ ДЛЯ ПРИЛОЖЕНИЯ ЗНАКОМСТВ

БАКАЛАВРСКАЯ РАБОТА

студента 4 курса 441 группы
направления 02.03.03 — Математическое обеспечение и администрирование информационных систем
факультета компьютерных наук и информационных технологий
Уталиева Султана Едильбаевича

Научный руководитель
ст.преп. кафедры ИиП _____ А. А. Казачкова

Заведующий кафедрой
к. ф.-м. н., доцент _____ Огнева М. В.

Саратов 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современные мобильные приложения всё чаще стремятся к персонализации пользовательского опыта. Одной из ключевых технологий в этой области являются рекомендательные системы, позволяющие адаптировать контент под индивидуальные предпочтения пользователей. Особенно важны такие системы в приложениях для знакомств, где точность рекомендаций напрямую влияет на успешность взаимодействий между пользователями.

В данной работе рассматривается задача построения рекомендательной системы для мобильного приложения знакомств. В отличие от большинства существующих решений, рекомендации в предлагаемой системе формируются не только на основе очевидных анкетных данных, но и на основе тестов-опросов, которые пользователь может пройти по желанию. Каждая карточка с вопросом предоставляет возможность ответить «да», «нет» или пропустить — что позволяет формировать тернарные признаки $(-1, 0, 1)$, лежащие в основе построения профиля предпочтений.

Основной целью дипломной работы является разработка рекомендательной системы, обеспечивающей релевантные и разнообразные рекомендации потенциальных партнеров на основе результатов тестирования. Для достижения этой цели решаются следующие задачи:

- анализ существующих подходов к построению рекомендательных систем в контексте приложений знакомств;
- формализация задачи рекомендаций с учетом специфики представления пользовательских данных;
- проектирование и реализация микросервисной архитектуры рекомендательной системы;
- исследование и разработка методов рекомендаций на основе коллаборативной фильтрации, эвристик и кластеризации;
- разработка подхода к оценке качества рекомендаций с использованием оффлайн-метрик и пользовательской обратной связи.

С точки зрения архитектуры, система реализована в виде набора взаимодействующих компонентов: мобильное приложение на Flutter, серверная часть на Java с использованием фреймворка Spring и отдельный рекомендательный микросервис на Python. Такое разделение позволяет гибко развивать систему и экспериментировать с алгоритмами без влияния на основной функционал

приложения.

Практическая значимость работы заключается в создании масштабируемого и расширяемого рекомендательного решения, которое может быть интегрировано в реальные приложения. Использование легких и интерпретируемых моделей делает систему доступной для внедрения даже в условиях ограниченных вычислительных ресурсов.

1 Анализ предметной области и существующих решений

1.1 Особенности рекомендательных систем

Рекомендательные системы (РС) являются неотъемлемой частью современных цифровых платформ, предоставляя пользователям персонализированные предложения продуктов, услуг или контента. Они находят широкое применение в различных областях, включая электронную коммерцию, стриминговые сервисы, социальные сети и онлайн-знакомства.

Основные функции рекомендательных систем включают:

— отсутствие информации о новых пользователях и объектах (проблема холодного старта);
— высокая разреженность пользовательско-объектных матриц;
— необходимость обеспечения справедливости и отсутствия предвзятости в рекомендациях [1].

Современные исследования направлены на преодоление этих ограничений, в том числе с использованием больших языковых моделей, методов объяснимого машинного обучения и расширения пользовательского контекста [2].

1.2 Обзор рекомендательных систем в сфере онлайн-знакомств

Онлайн-сервисы знакомств предъявляют особые требования к алгоритмам рекомендаций. Поскольку конечной целью является установление реального или виртуального контакта между людьми, системы должны максимально точно учитывать совместимость по широкому спектру признаков, при этом оставаясь достаточно лёгкими в вычислении и объяснимыми.

1.2.1 OkCupid

OkCupid начала с текстовых анкет и развила модель расчёта совместимости через детальные опросы. Пользователю предлагается до 400 вопросов, ответы на которые кодируются в вектор $q_i \in \{-1, 0, 1\}$, где значения означают «против», «безразлично» и «за» [3]. Система вычисляет попарную корреляцию ответов двух пользователей:

$$\mathrm{compat}(u, v) = \frac{\sum_i w_i \, \mathbf{1}(q_{u,i} = q_{v,i})}{\sum_i w_i},$$

где w_i — вес вопроса (определяется самим пользователем). Дальнейшее ранжирование учитывает активность и исторические данные о переписках. Такой

гибридный подход сочетает content- и collaborative-элементы и позволяет получать качественные рекомендации даже при отсутствии большого числа лайков или свайпов.

1.2.2 Tinder

Tinder применяет модель *reciprocal recommendation*: предлагаются только такие профили, которые с наибольшей вероятностью ответят взаимностью [4]. Основные компоненты алгоритма:

- базовый *рейтинговый Эло-подобный механизм*, где свайпы вправо увеличивают рейтинг профиля, влево уменьшают;
- измерение *скорости реакции* и *частоты использования* для оценки «живости» аккаунта;
- фильтрация неактивных и спорных профилей, чтобы не портить пользовательский опыт.

Исследования показывают, что такие меры уменьшают количество односторонних свайпов и повышают вероятность установления диалога между пользователями [5].

1.2.3 Bumble и Hinge

Bumble унаследовал от Tinder большинство алгоритмов, но ввёл правило, что первым сообщение отправляет только женщина. Это влияет на ранжирование: при прочих равных предпочтение отдается тем, кто реже получает безответные лайки и чаще инициирует общение [6]. Hinge выделяется принципом «создания общего контекста»: анализируются общие Facebook-друзья, интересы и местоположение. Рекомендации формируются через графовую модель, где вершины — пользователи, а рёбра взвешены по степени близости в социальных связях и ответам на три вступительных вопроса в профиле [7]. Это позволяет сегментировать рынок знакомств на локальные сообщества и повышает релевантность для тех, кто ищет знакомых «по кругу общения».

1.2.4 eHarmony

eHarmony фокусируется на психологической совместимости. Новый пользователь заполняет обширный личностный опросник, после чего строится профиль в латентном пространстве через метод факторизации матрицы сходств по тестовым шкалам (MMPI и Big Five) [8]. Дальнейшие рекомендации приме-

няют content-модуль для расширения круга потенциальных партнёров за счёт схожести по интересам и ценностям.

1.2.5 Coffee Meets Bagel и другие

Coffee Meets Bagel ежедневно генерирует «порции» (bagels) — ограниченный набор рекомендаций, отбираемых на основе коллаборативной фильтрации и эвристики «совместимых интересов» (аналог OkCupid, но с жёстким ограничением на количество предложений) [9]. Такой подход направлен на снижение перегрузки выбором и повышение качества каждого матча. Другие сервисы, например Plenty of Fish и Hily, экспериментируют с машинным зрением (анализ фотографий) и NLP (анализ биографий) для дополнительной фильтрации.

1.2.6 Общие тенденции

Современные платформы склоняются к гибридным решениям, объединяющим:

- анализ анкетных данных и опросов (content);
- collaborative filtering и reciprocal recommendation;
- эвристики активности и социального контекста;
- технологические новации (GNN, NLP, CV) для более глубокой семантической обработки.

Таким образом удаётся достичь баланса между качеством рекомендаций и вычислительными ресурсами, что особенно важно для мобильных приложений с высокой нагрузкой.

1.3 Особенности сбора и представления пользовательских признаков

В рекомендательных системах, применяемых в онлайн-знакомствах, центральную роль играет сбор и интерпретация информации о предпочтениях пользователей. Эти данные служат основой для построения персонализированных профилей и определения потенциально совместимых кандидатов. Анализ существующих решений в данной предметной области выявляет два крайних подхода к сбору информации:

- длинные анкеты с детализированными вопросами, характерные для платформ вроде eHarmony, обеспечивают богатое представление о пользователе, но требуют значительных усилий при заполнении и приводят к высокой доле отказов;

— минималистичные интерфейсы типа Tinder предлагают быструю оценку по принципу свайпа, обеспечивая высокую конверсию, но теряя глубину предпочтений.

В качестве компромиссного решения рассматриваются тернарные опросы, в которых каждый вопрос допускает три варианта реакции: положительную, отрицательную и нейтральную. Это позволяет выразить отношение к различным характеристикам без излишней нагрузки. Каждый ответ кодируется значением $q_i \in \{-1, 0, 1\}$, где i — номер вопроса. Такая шкала обеспечивает:

— компактность векторного представления профиля;
— возможность учитывать отсутствие чёткой позиции;
— поддержку различных методов машинного обучения и заполнения пропусков.

Результаты сбора формируются в разреженную матрицу $R = (q_{u,i})$, где u — пользователь, i — вопрос, а $q_{u,i}$ — реакция. Пропущенные значения соответствуют отсутствию взаимодействия. На практике возможно использование стратегий заполнения: нулями, средним значением по вопросу, либо построение модели, устойчивой к разреженности.

В дополнение к ответам могут быть включены и другие признаки:

— демографические данные — возраст (нормированный), пол (бинарная переменная);
— описания «о себе», конвертируемые в эмбеддинги с помощью языковых моделей;
— временные характеристики — длительность прохождения, количество пропусков, уверенность в ответах.

Особенность предметной области знакомств заключается в том, что каждый пользователь является одновременно субъектом и объектом рекомендаций. Это накладывает дополнительные требования к симметричности представлений и способности учитывать двустороннюю заинтересованность. Более того, успешность рекомендации здесь не сводится к лайку, а может быть оценена через цепочку взаимодействий: переписка, ответный интерес, встречи.

На этапе анализа задач были также выявлены следующие характеристики, которые следует учитывать при проектировании системы:

— необходимость работать с разреженными признаковыми матрицами и отсутствием части информации;

— поддержка холодного старта для новых пользователей и вопросов;
— ориентация на сравнение пользователей между собой, а не на ранжирование объектов фиксированной природы.

Таким образом, представление предпочтений в тернарной форме, дополненное демографией и эмбеддингами, формирует универсальную основу для построения профилей. Эти профили могут использоваться в различных методах рекомендации — от коллаборативной фильтрации до кластеризации и нейросетевых моделей.

2 Методы построения рекомендательной системы

2.1 Коллаборативная фильтрация

Коллаборативная фильтрация является одним из наиболее популярных подходов в рекомендательных системах, особенно в условиях, когда отсутствует явное описание объектов или пользователей. Основная идея заключается в том, что предпочтения пользователей могут быть предсказаны на основе поведения других пользователей с похожими вкусами.

Существует два основных типа коллаборативной фильтрации: на основе памяти (*memory-based*) и на основе модели (*model-based*). Первый подход использует метрики сходства между пользователями или объектами (например, косинусное расстояние, корреляцию Пирсона) и агрегирует оценки соседей. Второй — строит параметризованную модель на основе данных о взаимодействиях, чаще всего через факторизацию матрицы.

Пусть имеется матрица взаимодействий $R \in \mathbb{R}^{m \times n}$, где m — количество пользователей, n — количество объектов (например, анкет потенциальных партнёров). Элемент $r_{u,i}$ может обозначать бинарную оценку (лайк/не лайк), числовой рейтинг или иной сигнал предпочтения. Коллаборативная фильтрация предполагает, что в матрице присутствует скрытая структура, отражающая закономерности во вкусах пользователей.

Одним из распространённых методов является сингулярное разложение (SVD) или его модификации (например, Funk-SVD, ALS), при которых R аппроксимируется как

$$R \approx UV^\top,$$

где $U \in \mathbb{R}^{m \times k}$ и $V \in \mathbb{R}^{n \times k}$ содержат латентные векторы пользователей и объектов соответственно, а k — число латентных признаков. Значение $\hat{r}_{u,i} = U_u \cdot V_i^\top$ интерпретируется как предсказанная степень интереса пользователя u к объекту i.

Интересным обобщением является применение коллаборативной фильтрации к данным не только о лайках, но и о признаках, таких как ответы пользователей на опросы. В этом случае каждый пользователь представлен тернарным или категориальным вектором, а матрица $Q \in \{-1, 0, 1\}^{m \times p}$ (где p — число вопросов) также может быть факторизована аналогичным способом:

$$Q \approx U'Z^\top.$$

Полученные векторы могут использоваться для оценки схожести между пользователями, для восстановления пропущенных ответов или как источник признаков для гибридных моделей.

Коллаборативная фильтрация демонстрирует высокую эффективность при наличии большого количества пользовательских взаимодействий, однако страдает от проблемы холодного старта (new user/item problem) и может усиливать популярность одних и тех же объектов, снижая разнообразие рекомендаций.

2.2 Контентная фильтрация

Контентная фильтрация представляет собой один из классических подходов к построению рекомендательных систем. Основная идея заключается в том, чтобы рекомендовать объекты, схожие с теми, которые пользователь оценил положительно ранее, основываясь на характеристиках самих объектов. В отличие от коллаборативной фильтрации, здесь не учитываются предпочтения других пользователей.

Каждый объект описывается вектором признаков, которые могут быть бинарными, числовыми или категориальными. Пусть объект j представлен вектором признаков $\mathbf{x}_j \in \mathbb{R}^d$. Модель пользователя строится как агрегированное представление объектов, с которыми у него были положительные взаимодействия. Например, если пользователь i взаимодействовал с объектами $j_1, \ldots, j_k$, то вектор предпочтений можно получить как среднее:

$$\mathbf{p}_i = \frac{1}{k} \sum_{s=1}^{k} \mathbf{x}_{j_s}.$$

Рекомендации формируются на основе сходства между вектором предпочтений пользователя и векторами новых объектов. Чаще всего используется косинусное расстояние:

$$\mathrm{sim}(\mathbf{p}_i, \mathbf{x}_j) = \frac{\mathbf{p}_i^\top \mathbf{x}_j}{\|\mathbf{p}_i\| \cdot \|\mathbf{x}_j\|}.$$

Объекты с наибольшим значением sim включаются в топ рекомендаций.

Данный подход имеет ряд достоинств:

— высокая интерпретируемость: можно объяснить, почему был рекомендован тот или иной объект;

— независимость от количества других пользователей;

— устойчивость к проблеме холодного старта для объектов (если их описание доступно).

Однако контентная фильтрация имеет и ограничения:

— рекомендации ограничиваются областью уже проявленных интересов;
— трудно учитывать сложные зависимости между признаками;
— качество зависит от полноты и выразительности признакового описания.

Для повышения гибкости могут применяться методы машинного обучения. Например, обучающая выборка может включать пары $(\mathbf{x}_j, y_{ij})$, где y_{ij} — бинарная переменная, указывающая наличие положительного отклика со стороны пользователя i на объект j. На этой основе можно обучить логистическую регрессию, SVM или градиентный бустинг, предсказывающий вероятность интереса к новому объекту.

Также возможны гибридные модели, объединяющие контентную и коллаборативную фильтрацию. Например, контентные признаки могут использоваться для регуляризации матричной факторизации или служить входом для нейронных сетей. Такие подходы позволяют улучшить обобщающую способность модели и преодолеть узость интересов, характерную для чисто контентной фильтрации.

Контентные методы особенно полезны в системах, где объекты имеют чётко выраженные признаки: тексты, категории, изображения, ответы на тесты или анкеты. При наличии информативного описания они позволяют получать рекомендации уже на самых ранних этапах использования системы, что делает их важным компонентом гибридных решений.

2.3 Эвристики: совпадения по ответам, популярность, фильтры

Эвристические методы в рекомендательных системах основаны на наборе простых правил и предположений, позволяющих быстро и эффективно формировать рекомендации. Несмотря на относительную простоту, такие подходы остаются актуальными, особенно в условиях ограниченности данных или требований к объяснимости.

Одним из базовых эвристических подходов является сравнение пользователей по их ответам на вопросы анкет или опросов. Если ответы представлены в тернарной шкале (например, -1 — несогласие, 0 — нейтрально, 1 — согласие), то схожесть между пользователями можно оценить по доле совпадающих

ответов:

$$\mathrm{sim}(u, v) = \frac{1}{|P|} \sum_{i \in P} \mathbf{1}(q_{u,i} = q_{v,i}),$$

где P — множество вопросов, на которые оба пользователя дали ответ. Этот подход применим как в системах знакомств, так и в других областях, где важна совместимость взглядов, предпочтений и интересов.

Другим эвристическим приёмом является использование показателя популярности. Объекты (например, профили или товары), получившие наибольшее количество положительных оценок, могут предлагаться новым пользователям в качестве стартовых рекомендаций. Популярность может быть нормирована, например:

$$\mathrm{pop}(i) = \frac{\text{число лайков объекта } i}{\text{максимальное число лайков среди всех объектов}}.$$

Популярные рекомендации часто дополняются фильтрацией по демографическим и другим признакам, таким как возраст, пол, географическое местоположение, язык, наличие общих интересов и т. д.

Такие фильтры применяются до или после основного ранжирования и позволяют исключить очевидно нерелевантные варианты. Например, пользователь, заинтересованный только в кандидатах определённого возраста или пола, должен получать только соответствующие предложения.

Также может применяться эвристика совпадения по ключевым признакам. Если в профиле пользователя указаны предпочтения (например, любимые фильмы, занятия, взгляды), система может искать совпадения с другими профилями и ранжировать их по количеству совпавших интересов.

Комбинирование эвристик позволяет построить гибкую систему, способную адаптироваться к условиям отсутствия данных, начальной загрузки, а также повысить обоснованность рекомендаций. При этом эвристические методы легко интерпретируемы, что важно в чувствительных сферах, таких как онлайн-знакомства или подбор персонала.

2.4 Кластеризация (k-means, DBSCAN) и применение в рекомендациях

Кластеризация — это метод обучения без учителя, направленный на группировку объектов в кластеры таким образом, чтобы элементы одного кластера

были похожи друг на друга и отличались от элементов других кластеров. В контексте рекомендательных систем кластеризация применяется для:

— сегментирования пользователей (или объектов) по интересам, поведению или признакам;
— повышения масштабируемости рекомендаций за счёт ограничения поиска релевантных кандидатов внутри кластера;
— выявления нишевых предпочтений и персонализированных паттернов.

Одним из наиболее популярных алгоритмов является *k-means*. Он принимает на вход число кластеров k и итеративно минимизирует внутрикластерную дисперсию:

$$\sum_{j=1}^{k} \sum_{x_i \in C_j} \|x_i - \mu_j\|^2,$$

где C_j — кластер, а μ_j — его центр. Алгоритм эффективен при компактных и сферических кластерах, но чувствителен к выбору k и неустойчив к выбросам.

Для более гибкой кластеризации используется алгоритм *DBSCAN* (Density-Based Spatial Clustering of Applications with Noise). Он группирует точки по плотности: кластером считается связная по плотности область, где каждая точка имеет хотя бы $minPts$ соседей в пределах радиуса ε. DBSCAN способен находить кластеры произвольной формы и автоматически игнорирует шум (выбросы), что делает его особенно полезным в гетерогенных данных.

В рекомендательных системах кластеризация применяется по-разному:

— На пространстве пользователей: сегментация по поведенческим или опросным признакам позволяет формировать кластеры пользователей с похожими предпочтениями. Рекомендации для нового пользователя можно извлекать из наиболее близкого кластера.
— На пространстве объектов: группировка товаров, фильмов, анкет и пр. по тематике, стилю или целевой аудитории позволяет адаптировать рекомендации к интересам пользователя.
— В латентных пространствах: кластеризация векторов после факторизации (например, в SVD или autoencoder-подходах) даёт более сжатое и семантически значимое представление.

Кластеризация также применяется для визуализации и анализа структуры пользовательской базы, выявления целевых групп и построения тематических подборок. Её эффективность во многом зависит от выбора признаков и мас-

штабов данных, поэтому нередко она используется в комбинации с другими методами.

2.5 Стратегии холодного старта

Проблема холодного старта возникает, когда система не располагает достаточной информацией о пользователях или объектах, чтобы формировать персонализированные рекомендации. Выделяют два основных сценария: появление нового пользователя и добавление нового объекта (например, анкеты).

Для новых пользователей могут применяться следующие подходы:

- заполнение вступительных тестов или анкет — позволяет собрать первичные признаки и использовать их при формировании рекомендаций;
- использование демографических данных — рекомендации подбираются на основе поведения пользователей с аналогичными характеристиками (возраст, пол, география и т.д.);
- показ популярных объектов — временная стратегия, при которой пользователю демонстрируются анкеты с высокой оборачиваемостью, что помогает быстрее сформировать профиль предпочтений.

При появлении новых объектов, которые ещё не получили откликов, возможны такие меры:

- временное повышение приоритета в выдаче — например, показ новым или активным пользователям для ускоренного накопления статистики;
- использование признаков схожести — на основе анкетных данных, внешности (в случае CV), текста описания (в случае NLP) или ответов на вопросы;
- размещение в релевантных сегментах — объект может быть временно включён в выдачу по кластерам, в которые он потенциально попадает по признаковому пространству.

Кроме того, существуют универсальные стратегии, применимые и к новым пользователям, и к новым объектам:

- инициализация признаков с помощью доступных внешних данных — анкет, биографий, метаинформации;
- использование гибридных моделей, сочетающих элементы content-based и коллаборативной фильтрации — это снижает чувствительность к отсутствию истории;

— активное обучение — выбор контента, максимально полезного для уточнения предпочтений, что позволяет за минимальное число взаимодействий улучшить качество рекомендаций.

Проблема холодного старта наиболее критична для систем, основанных на коллаборативной фильтрации, поскольку они требуют исторических данных о взаимодействии пользователей с объектами. Поэтому многие современные решения строятся с использованием дополнительных эвристик и предварительной инициализации признаков, что позволяет обеспечить устойчивость системы на ранних этапах использования.

2.6 Методы глубокого обучения в рекомендательных системах

Развитие нейросетевых архитектур оказало существенное влияние на область рекомендательных систем. Благодаря способности извлекать сложные латентные зависимости из разнородных данных, модели глубокого обучения применяются для повышения качества рекомендаций как в традиционных задачах (например, предсказание рейтингов), так и в более сложных сценариях, включая мультимодальные рекомендации, учет временной динамики и персонализацию на основе контекста.

Одним из первых направлений стало расширение классической матричной факторизации с помощью нейронных сетей. Вместо простой линейной факторизации матрицы взаимодействий $R \in \mathbb{R}^{m \times n}$, где m — количество пользователей, n — количество объектов, и R_{ij} — факт взаимодействия, используются обучаемые эмбеддинги и нелинейные функции активации. Примером такой модели является Neural Collaborative Filtering (NCF), предложенная в [**?**]. В NCF пары эмбеддингов $(\mathbf{u}_i, \mathbf{v}_j)$ передаются через многослойный перцептрон:

$$\hat{r}_{ij} = \mathrm{MLP}([\mathbf{u}_i, \mathbf{v}_j]),$$

где $[\cdot, \cdot]$ обозначает конкатенацию векторов. Модель обучается по функции потерь, например, бинарной кросс-энтропии в задаче предсказания лайка/дизлайка.

Другое направление связано с использованием рекуррентных и трансформерных архитектур для моделирования последовательности взаимодействий. Так называемые sequence-based recommenders учитывают порядок взаимодействий пользователя с объектами. Например, модель GRU4Rec [**?**] применяет

Gated Recurrent Unit (GRU) для обработки истории действий пользователя. Базовая идея заключается в следующем: пусть $x_1, x_2, \ldots, x_T$ — последовательность взаимодействий, тогда на каждом шаге рассчитывается скрытое состояние h_t:

$$h_t = \mathrm{GRU}(x_t, h_{t-1}),$$

и предсказывается следующий элемент x_{t+1} с помощью softmax-слоя.

Для учёта более длинных зависимостей и параллельной обработки была предложена модель SASRec [**?**], основанная на self-attention. В отличие от рекуррентных моделей, здесь используется позиционно-кодированная последовательность эмбеддингов, проходящая через слои трансформера. Это позволяет учитывать контекст всех предыдущих действий при выборе следующей рекомендации.

Особое место занимают графовые нейронные сети, применяемые для моделирования взаимодействий в виде графов. В Graph Convolutional Matrix Completion (GC-MC) [**?**], каждый пользователь и объект представляются вершинами, соединёнными ребром при наличии взаимодействия. Представления вершин обновляются по правилу:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}(i)} \frac{1}{c_{ij}} W^{(l)} \mathbf{h}_j^{(l)}\right),$$

где $\mathcal{N}(i)$ — соседи вершины i, $W^{(l)}$ — матрица весов на l-м слое, c_{ij} — коэффициент нормализации. Такой подход позволяет учитывать структуру взаимодействий и взаимосвязь объектов, что особенно актуально для задач, где важны не только пользовательские предпочтения, но и социальные или контекстные связи.

Ещё одно активно развивающееся направление — мультимодальные рекомендации. Здесь помимо взаимодействий учитываются дополнительные признаки, такие как текст описания, изображения, аудио. Для обработки таких данных применяются CNN, BERT и другие специализированные архитектуры. В модели VBPR [**?**] визуальные признаки изображений используются для расширения латентного пространства:

$$\hat{r}_{ij} = \mathbf{u}_i^\top \mathbf{v}_j + \mathbf{u}_i^\top E x_j,$$

где x_j — визуальный вектор, полученный из изображения объекта, E — обучаемая матрица проекции.

Основные преимущества методов глубокого обучения:

— способность моделировать сложные нелинейные зависимости между пользователями и объектами;
— возможность использовать разнородные источники информации;
— высокая гибкость и расширяемость архитектур.

Тем не менее, существуют и ограничения:

— высокая требовательность к вычислительным ресурсам;
— потребность в большом объёме размеченных данных;
— трудность интерпретации и объяснимости результатов.

Таким образом, методы глубокого обучения открывают широкие перспективы для построения более точных и персонализированных рекомендательных систем, особенно в условиях, когда доступны дополнительные источники информации и достаточно ресурсов для обучения. Однако выбор таких подходов должен быть обоснован задачами проекта, размером аудитории и доступной инфраструктурой.

2.7 Гибридные подходы

Гибридные рекомендательные системы объединяют преимущества различных методов, включая коллаборативную фильтрацию, контентную фильтрацию и эвристические алгоритмы. Такая интеграция позволяет компенсировать слабые стороны отдельных подходов и достичь более высокой точности, устойчивости к холодному старту и разнообразия рекомендаций.

Существуют разные стратегии гибридизации:

— объединение выходов нескольких моделей (late fusion);
— комбинирование признаков на входе одной модели (early fusion);
— использование одного подхода в качестве фильтра, а другого — для ранжирования.

Один из классических примеров — модель, в которой одновременно учитываются схожесть пользователей (коллаборативная составляющая) и сходство объектов (контентная составляющая). Пусть r_{ui} — предсказанная оценка пользователя u для объекта i. Тогда комбинированная формула может иметь следующий вид:

$$r_{ui} = \alpha \cdot r_{ui}^{\text{collab}} + (1 - \alpha) \cdot r_{ui}^{\text{content}},$$

где r_{ui}^{collab} — предсказание по коллаборативной модели, r_{ui}^{content} — результат контентного ранжирования, а $\alpha \in [0, 1]$ — параметр, регулирующий вклад каждой части.

Другой подход основан на поэтапной фильтрации. Например, можно сначала применить контентную фильтрацию для предварительного отбора релевантных объектов, а затем выполнить коллаборативное ранжирование по пользовательским лайкам или оценкам. Такой двухшаговый процесс снижает вычислительную нагрузку и повышает релевантность.

Гибридные системы особенно полезны в следующих сценариях:

— наличие разреженной матрицы пользовательских взаимодействий, при этом имеются структурированные признаки объектов;
— необходимость рекомендаций для новых пользователей или новых объектов;
— желание учитывать не только историю взаимодействий, но и семантическое содержание;
— ориентация на объяснимость модели и возможность интерактивной настройки предпочтений.

В последние годы широкое распространение получили модели, встраивающие оба подхода в общую латентную структуру. Например, в модели Factorization Machines (FM) и её нейронных расширениях (Neural FM, DeepFM) признаки пользователей и объектов подаются в общий обучаемый слой, позволяющий учитывать как контентную, так и взаимодействующую информацию. Обозначим входной вектор признаков как $\mathbf{x}$, тогда предсказание в FM-модели имеет вид:

$$\hat{y}(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j,$$

где $\mathbf{v}_i$ — латентный вектор i-го признака. Такая модель способна улавливать взаимодействия между признаками без явного задания структуры.

Гибридные методы на практике показывают высокую гибкость и адаптивность, особенно в условиях изменяющихся предпочтений и ограниченных пользовательских данных. Они успешно применяются в рекомендательных системах

крупных платформ (Netflix, YouTube, LinkedIn), где важно сочетать поведенческие паттерны с контекстной и персонализированной информацией.

Совокупность перечисленных подходов делает гибридные методы универсальным инструментом, который можно адаптировать под особенности конкретной предметной области, в том числе в сфере онлайн-знакомств.

3 Архитектура приложения и формализация задачи

3.1 Описание мобильного приложения

Мобильное приложение, разработанное на платформе Flutter, представляет собой современный клиент для сервиса знакомств, ориентированный на сбор признаков пользователей и формирование персонализированных рекомендаций. Интерфейс построен вокруг свайп-механики, однако дополняется функциональностью, позволяющей учитывать более тонкие предпочтения, взаимодействовать с другими пользователями, а также заполнять собственный профиль.

После запуска пользователь попадает на экран входа в аккаунт (Рисунок 1). Этот экран обеспечивает точку входа в систему и привязку действий к конкретному идентификатору пользователя.

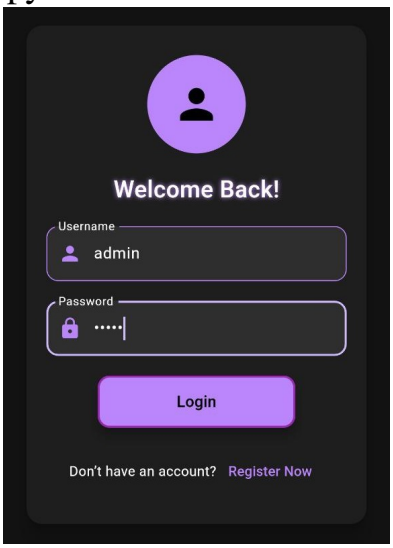

Рисунок 1 – Экран входа в аккаунт.

Главным элементом пользовательского опыта является блок рекомендаций. Здесь отображается список потенциальных партнёров, отсортированных по метрике сходства, которая принимает значения от -100 до 100. Эта величина отражает уровень соответствия интересов между текущим пользователем и кандидатом, вычисляемый на основе тернарных признаков.

Для каждого пользователя можно:

— перейти в профиль для просмотра подробной информации;

— отправить заявку на чат, если он кажется интересным.

Пример данного экрана представлен на Рисунке 2.

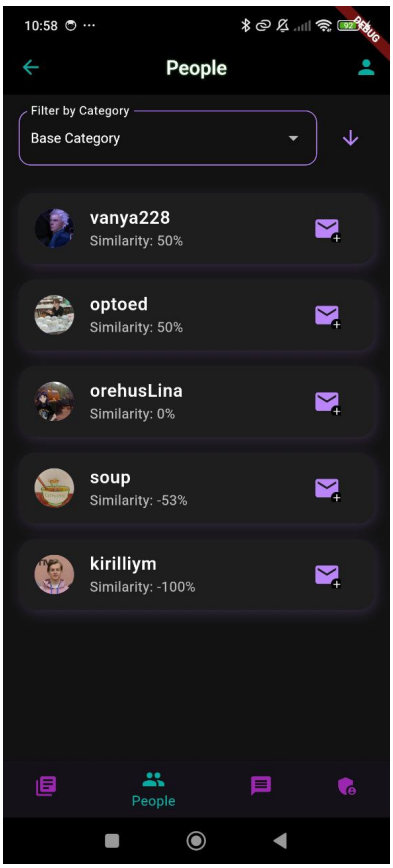

Рисунок 2 – Экран с рекомендациями пользователей

Сбор признаков осуществляется с помощью системы тестов, оформленных в виде свайп-карточек. Каждый тест содержит вопросы на определённую тему, например, предпочтения в музыке. Пользователь может смахнуть карточку вправо (согласие), влево (несогласие) или вверх (пропуск), что формирует тернарный вектор ответов $q_i \in \{-1, 0, 1\}$. На карточке указывается, в какую сторону нужно свайпать, чтобы выбрать тот или иной ответ.

Интерфейс свайпа организован так, чтобы пользователь мог быстро и интуитивно отвечать на вопросы, минимизируя нагрузку и повышая вовлечённость. Сама механика свайпа показана на Рисунке 3.

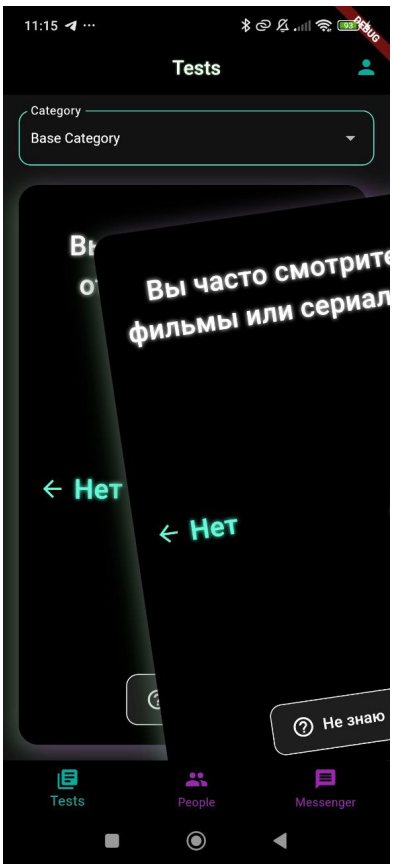

Рисунок 3 – Пример карточки теста

Профиль пользователя позволяет редактировать личную информацию, менять фотографию, а также переходить в настройки или выйти из аккаунта. Этот экран представлен на Рисунке 4.

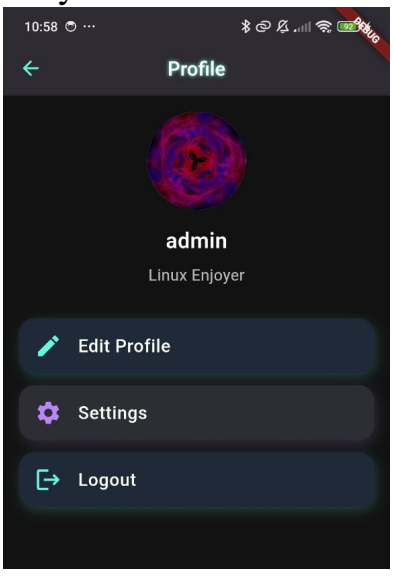

Рисунок 4 – Экран профиля пользователя

Для администраторов предусмотрена возможность пополнять базу вопросов напрямую из приложения. Добавление нового вопроса в выбранную категорию осуществляется через специальный интерфейс (Рисунок 5).

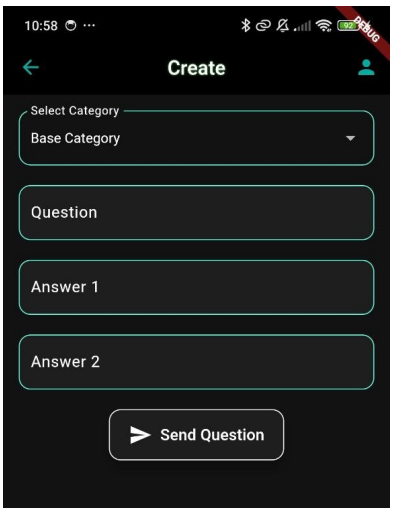

Рисунок 5 – Интерфейс добавления вопроса администратором

Когда пользователь получает входящую заявку на общение, она отображается на отдельном экране. Здесь доступен выбор: принять или отклонить приглашение. Это помогает формировать согласованные чаты на основе взаимного интереса (Рисунок 6).

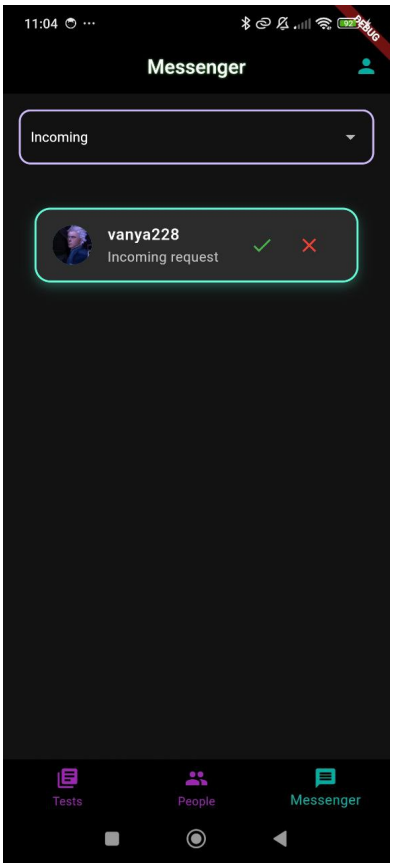

Рисунок 6 – Входящие заявки на чат

Чат представляет собой полноценный мессенджер, позволяющий отправлять текстовые сообщения и прикреплять изображения. Также поддерживается возможность удаления диалогов, что даёт пользователям контроль над своим взаимодействием (Рисунок 7).

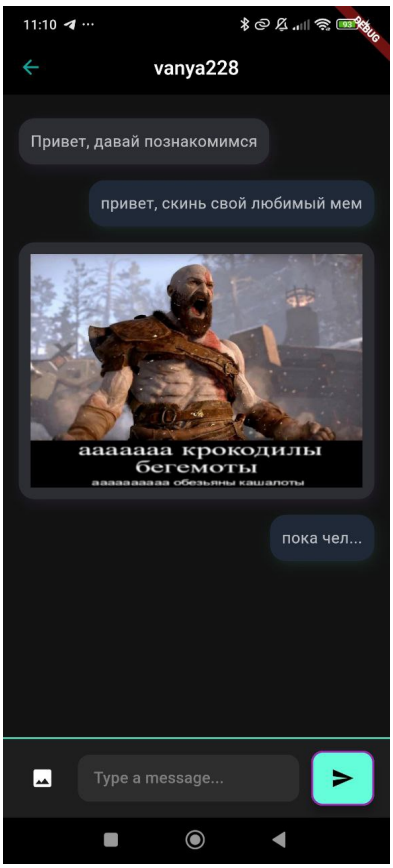

Рисунок 7 – Экран чата с поддержкой отправки изображений

Таким образом, мобильное приложение реализует весь цикл взаимодействия: от сбора предпочтений до установления контактов между пользователями. При этом особое внимание уделено простоте интерфейса и гибкости сбора признаков, что делает возможным последующее применение методов рекомендаций различной природы.

3.2 Архитектура клиент-серверной системы

Система построена по принципу микросервисной архитектуры с чётким разделением ответственности между клиентским приложением, серверной частью и модулем рекомендательной системы. Такое проектирование обеспечивает масштабируемость, возможность независимого обновления компонентов, а также гибкость в экспериментировании с различными алгоритмами на стороне рекомендаций.

В основе клиентской части лежит кроссплатформенное мобильное приложение, разработанное с использованием фреймворка Flutter. Оно обеспечивает интерактивный интерфейс для пользователя, включая регистрацию, прохождение тестов, просмотр рекомендаций, обмен сообщениями, а также базовое редактирование профиля. Приложение взаимодействует с серверной частью по

REST-протоколу, отправляя и получая данные в формате JSON.

Серверная часть реализована на Java с использованием фреймворка Spring Boot и выступает в роли API-шлюза и основной точки обработки бизнес-логики. Она обеспечивает:

— аутентификацию и авторизацию пользователей;
— хранение информации о профилях, вопросах и ответах;
— маршрутизацию данных между компонентами;
— взаимодействие с системой рекомендаций через HTTP-интерфейс.

База данных (используется PostgreSQL) служит хранилищем для всех структурированных сущностей: пользователей, результатов тестов, заявок на чат и переписок. Для вложений (изображений и аватаров) применяется объектное хранилище, совместимое с S3-протоколом. Кроме того, используется Redis как in-memory кеш для ускорения обработки часто запрашиваемых данных, например, при повторном открытии профиля или списка входящих заявок.

Отдельный компонент системы — сервис рекомендаций, реализованный на Python и развёрнутый как изолированный микросервис. Он не входит напрямую в периметр Java-приложения, но взаимодействует с ним по API. Рекомендательный модуль получает от бэкенда вектор признаков пользователя и возвращает отсортированный список идентификаторов наиболее подходящих партнёров, снабжённый числовой метрикой схожести. Такие предсказания строятся на основе тернарных ответов $q_{u,i} \in \{-1, 0, 1\}$ и, при необходимости, дополнительно учитывают демографические параметры.

С архитектурной точки зрения, система логически разделяется на три уровня:

1. Клиентский уровень. Представлен Flutter-приложением, которое отвечает за интерактивный ввод, навигацию, отображение данных и обратную связь с пользователем. Также приложение реализует базовую валидацию данных (например, обязательность полей при регистрации) до их отправки на сервер.
2. Уровень бизнес-логики. Серверное приложение на Spring Boot, реализующее REST-контроллеры, обработку авторизации (через JWT), управление данными, транзакции и обработку ошибок. Оно взаимодействует с БД, кешем и внешними сервисами, включая рекомендательную систему.
3. Аналитический уровень. Python-микросервис, реализующий методы кол-

лаборативной фильтрации и другие эвристики. Имеет собственный pipeline подготовки данных (например, регулярное построение матрицы признаков и SVD-декомпозиции) и отвечает на запросы по HTTP.

При такой организации достигаются следующие преимущества:

— масштабирование микросервисов независимо друг от друга;
— возможность A/B-тестирования алгоритмов рекомендаций без изменения основного приложения;
— снижение времени отклика за счёт кеширования;
— удобство сопровождения: изменения в логике рекомендации не требуют вмешательства в клиент или основное серверное приложение.

Каждое взаимодействие клиента и сервера документировано через OpenAPI, что упрощает генерацию SDK и отладку запросов.

Безопасность обеспечивается на нескольких уровнях: JWT-токены для пользователей а также проверка полномочий при доступе к ресурсам.

Таким образом, архитектура системы позволяет гибко расширять функциональность, проводить независимую разработку и тестирование модулей, а также адаптировать систему под рост пользовательской базы без полной переработки существующего кода.

3.3 Формализация задачи рекомендаций

Целью рекомендательной системы, интегрированной в мобильное приложение, является предоставление пользователю персонализированного списка наиболее подходящих партнёров. Формально эта задача рассматривается как задача ранжирования кандидатов на основе оценки их релевантности конкретному пользователю.

Пусть $U = \{u_1, u_2, \ldots, u_m\}$ — множество пользователей системы. Каждый пользователь u_i проходит один или несколько тестов, состоящих из множества вопросов $Q = \{q_1, q_2, \ldots, q_n\}$. Ответы пользователя на тесты кодируются вектором:

$$\mathbf{v}_i = (a_{i1}, a_{i2}, \ldots, a_{in}), \quad a_{ij} \in \{-1, 0, 1\}$$

где

— $a_{ij} = 1$ означает положительный ответ на вопрос q_j;
— $a_{ij} = -1$ — отрицательный;
— $a_{ij} = 0$ — отсутствие ответа.

Таким образом, каждый пользователь представлен тернарным вектором фиксированной размерности n, допускающим пропуски.

Дополнительно профиль пользователя может включать следующие признаки:

— пол $s_i \in \{0, 1\}$;

— возраст $a_i \in \mathbb{R}$ (предварительно нормированный);

— эмбеддинг текстового описания профиля $\mathbf{d}_i \in \mathbb{R}^k$.

Полный вектор признаков пользователя обозначим как:

$$\mathbf{x}_i = (\mathbf{v}_i, s_i, a_i, \mathbf{d}_i)$$

Задача рекомендательной системы заключается в построении функции релевантности $\text{score}(u_i, u_j)$, измеряющей степень соответствия между пользователями u_i и u_j. Для каждого пользователя u_i система должна вернуть упорядоченный список пользователей $R_i \subseteq U \setminus \{u_i\}$, отсортированный по убыванию релевантности:

$$\text{score}(u_i, u_j) > \text{score}(u_i, u_k) \Rightarrow u_j \text{ выше в списке рекомендаций, чем } u_k$$

В качестве функции $\text{score}(\cdot, \cdot)$ могут использоваться следующие подходы:

— косинусное сходство между векторами признаков:

$$\text{score}(u_i, u_j) = \frac{\langle \mathbf{x}_i, \mathbf{x}_j \rangle}{\|\mathbf{x}_i\| \cdot \|\mathbf{x}_j\|}$$

— скалярное произведение эмбеддингов, полученных через матричную факторизацию:

$$\text{score}(u_i, u_j) = \langle f(u_i), g(u_j) \rangle$$

— вероятностная оценка интереса:

$$\text{score}(u_i, u_j) = \mathbb{P}(u_i \text{ поставит лайк } u_j)$$

Обучение системы может проводиться по историческим данным о взаимодействии между пользователями. В базовой реализации предполагается использование метода коллаборативной фильтрации на основе матричной факторизации, где модель обучается на матрице оценок $R \in \mathbb{R}^{m \times m}$, где $r_{ij} = 1$, если

пользователь u_i поставил лайк пользователю u_j, и $r_{ij} = 0$ в противном случае.

Оптимизационная цель может формулироваться как максимизация точности попадания релевантных пользователей в топ-k:

$$\max \frac{1}{|U|} \sum_{i=1}^{m} \frac{|R_i^{\text{true}} \cap R_i^{\text{pred}}(k)|}{k}$$

где R_i^{true} — множество пользователей, которым u_i поставил лайк, а $R_i^{\text{pred}}(k)$ — топ-k рекомендаций для u_i.

В перспективе возможно включение дополнительных источников информации, таких как временная динамика лайков, контекст использования приложения или текстовые описания. Это позволит перейти к более сложным моделям, основанным на нейросетевых архитектурах или графовых представлениях.

1 Jin, D. A survey on fairness-aware recommender systems / D. Jin, L. Wang, H. Zhang, Y. Zheng, W. Ding, F. Xia, S. Pan // arXiv preprint arXiv:2306.00403. — 2023.

2 Recommender systems in the era of large language models (llms) / Z. Zhao, W. Fan, J. Li, Y. Liu, X. Mei, Y. Wang, Z. Wen et al. // arXiv preprint arXiv:2307.02046. — 2023.

3 Hitsch, G. J. What makes you click?—mate preferences in online dating / G. J. Hitsch, A. Horta?su, D. Ariely // Quantitative Marketing and Economics. — 2010. — T. 8, no. 4. — Pp. 393–427.

4 Courtois, C. Cracking the tinder code: An experience sampling approach to the dynamics and impact of platform governing algorithms / C. Courtois, E. Timmermans // Journal of Computer-Mediated Communication. — 2018. — T. 23, no. 1. — Pp. 1–16.

5 Pidoux, J. Online Dating Quantification Practices: A Human–Machine Learning Process: Ph.D. thesis / ?cole Polytechnique F?d?rale de Lausanne. — 2021.

6 Jennings, R. The tinder algorithm, explained / R. Jennings // Vox. — 2019. https://www.vox.com/2019/2/7/18210998/tinder-algorithm-swiping-tips-dating-app-science.

7 Bruch, E. E. Aspirational pursuit of mates in online dating markets / E. E. Bruch, M. E. J. Newman // EPJ Data Science. — 2018. — T. 7, no. 1. — Pp. 1–14.

8 Lefebvre, H. Psychometric matching and the evolution of eharmony's recommendation engine / H. Lefebvre, Y. Dallal // Journal of Personality and Social Psychology. — 2022. — T. 123, no. 5. — Pp. 912–926.

9 Lee, A. Limited choice, better matches: The coffee meets bagel approach // Proceedings of the 2021 ACM Conference on Recommender Systems. — ACM, 2021. — Pp. 345–350.