Санкт–Петербургский государственный университет

Шумов Дмитрий Русланович

Выпускная квалификационная работа

Построение гибридной рекомендательной системы на основе коллаборативных алгоритмов и машинного обучения

Уровень образования: бакалавриат
Направление 01.03.02 «Прикладная математика и информатика»
Основная образовательная программа СВ.5005.2015 «Прикладная математика, фундаментальная информатика и программирование»
Профиль «Математическое обеспечение вычислительных машин, комплексов и компьютерных сетей.»

Научный руководитель:
сташий преподаватель, кафедра технологии программирования,
Стученков Александр Борисович

Рецензент:
профессор, кафедра информационных систем,
д.ф. - м.н. Матросов Александр Васильевич

Санкт-Петербург
2021 г.

Содержание

Введение 3

Постановка задачи 3

Глава 1. Рекомендательные системы 4

§1. Рекомендательные системы на основе коллаборативной фильтрации 4

1.1 Memory-based 5

1.2 Model-based 7

1.2.1 Алгоритмы разложения матриц в контексте рекомендательных систем 7

1.2.2 Нейронные сети для рекомендаций 13

§2. Рекомендательные системы на основе содержимого 15

§3. Гибридные рекомендательные системы 18

Глава 2. Программная реализация 19

§1. Выбор инструментов разработки 19

§2. Структура проекта 20

Глава 3. Эффективность рекомендаций 22

§1. Метрики оценивания 22

1.1 Регресионные метрики 22

1.2 Метрики классификации 23

1.3 Метрики качества ранжирования 25

§2. Сравнение алгоритмов 27

2.1 Memory-based 27

2.2 Model-based 29

2.3 Content-based 33

2.4 Hybrid 34

Список литературы 35

Введение

В условиях избытка информации, характерного для нашего времени, существенной является проблема её отбора. Поэтому актуальной является разработка инструментов, позволяющих упростить эту задачу. Рекомендательные системы призваны облегчить жизнь конечного пользователя, подстроившись под его интересы. Они нашли широкое применение в самых различных сферах: онлайн-торговле, стриминговых сервисах, поисковых системах, рекламе. Бизнес использует их для повышения привлекательности своих ресурсов или для увеличения продаж. В основу рекомендательных систем легли алгоритмы коллаборативной фильтрации - простые, но допускающие большое число модификаций и комбинирование с другими алгоритмами.

Постановка задачи

Цель работы: разработать программное обеспечение для рекомендаций пользователю фильмов.

Можно выделить следующие задачи для достижения цели работы:

- Исследовать популярные способы построения рекомендательных систем
- Рассмотреть различные модификации
- Выделить наиболее эффективный для конкретной области подход
- Реализовать рекомендательную систему, использующую выбранный алгоритм и оценить качество её работы
- Проанализировать полученные результаты

Глава 1. Рекомендательные системы

Рекомендательные системы — компьютерные программы, используемые в попытке предсказать какому объекту пользователь отдаст своё предпочтение или какой "рейтинг" поставит. Эти системы персонализируют наше взаимодействие с сетью, подсказывая что посмотреть, что купить, что послушать, с кем подружиться, что почитать и т.д. Для этого они анализируют наше взаимодействие с различными сервисами и выделяют шаблоны поведения, а также объекты, с которыми мы взаимодействуем. Существенным для рекомендательных систем является накопление знаний об активности пользователей или свойствах объектов. Самые базовые имплементации основываются на предположении о том, что людям понравятся вещи, похожие на те, что им уже нравятся, а также вещи, которые нравятся людям с похожим вкусом.

Рекомендательные системы можно разбить на 3 категории:

- На основе содержимого — модель, которая использует множество свойств объекта для рекомендации объектов с похожими характеристиками.
- На основе коллаборативной фильтрации — модель, которая учитывает историю взаимодействия пользователя с сервисом (купленные товары, прослушанная музыка и т.п.), а также поведение похожих пользователей, а затем использует эту информацию для рекомендации объектов, которые могут заинтересовать пользователя.
- Гибридные системы — модель, сочетающая в себе два предыдущих подхода.

§1 Рекомендательные системы на основе коллаборативной фильтрации

Коллаборативная фильтрация использует данные о поведении пользователей. Данные о взаимодействии пользователей с контентом могут определяться оценками, которые пользователь поставил, и действиями, которые совершил (например, просмотрел карточку товара). На основе этих данных

система может предсказать, насколько сильно пользователю понравится объект, с которым он ещё не взаимодействовал. Техники коллаборативной фильтрации можно разделить на 2 типа:

- Memory-based — основывается на вычислении "схожести" пользователей или объектов их интереса
- Model-based — предполагает использование алгоритмов машинного обучения для предсказания пользовательского отношения к объектам

1.1 Memory-based

Эта техника предполагает два подхода: user-based и item-based. User-based подход предполагает вычисление "схожести" между пользователями, чтобы рекомендовать конкретному человеку то, что нравится похожим на него людям. Item-based подход предполагает вычисление "схожести" между объектами, чтобы рекомендовать конкретному человеку объекты, похожие на те, которыми он обычно интересуется. Схожесть в свою очередь вычисляется, исходя из взаимосвязей пользователей и объектов. Результатом работы является матрица предсказанных оценок, в которой построчно стоят вектора оценок для конкретного пользователя (user-based) или вектора оценок для конкретного объекта (item-based).

Вычисление схожести

В первых работах по коллаборативной фильтрации обычно использовалась корреляция Пирсона [1]:

$$sim(u,v) = \frac{\sum_{i \in I_u \cap I_v} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in I_u \cap I_v} (r_{ui} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_u \cap I_v} (r_{vi} - \bar{r}_v)^2}}, \quad (1)$$

где I_u и I_v — множество оценок, поставленных пользователями u и v соответственно, r_{ui} — оценка пользователя u объекту i, $\bar{r}_u$ — среднее арифметическое оценок пользователя u

У такого вычисления есть существенный недостаток — оно отбрасывает объекты, оценки для которых предоставил один пользователь, но не предо-

ставил другой. Это может привести к тому, что два разных пользователя, одинаково оценившие несколько одних и тех же объектов, но имеющие много других оценок, будут иметь очень высокий коэффициент схожести. Способ исправить это был предложен в [2]. Он заключается в том, чтобы домножить схожесть на $\frac{\min(|I_u \cap I_v|, n)}{n}$, тем самым уменьшая её, если у пользователей менее n общих объектов. Частно применимая поправка, но позволяет улучшить результат.

Также для подсчёта схожести предлагалось косинусное сходство [3]:

$$sim(u, v) = \frac{r_u \cdot r_v}{\|r_u\| \|r_v\|},$$

Однако, лучше всего себя показало центрированное косинусное сходство очень похожее на (1):

$$sim(u, v) = \frac{\sum_{i \in I_u \cup I_v} \hat{r_{ui}} \hat{r_{vi}}}{\sqrt{\sum_{i \in I_u \cup I_v} \hat{r_{ui}}^2} \sqrt{\sum_{i \in I_u \cup I_v} \hat{r_{vi}}^2}},$$

где

$$\hat{r_{ui}} = \begin{cases} 0, & r_{ui} = 0 \\ r_{ui} - \bar{r}_u, & r_{ui} \neq 0 \end{cases}$$

Которое, ведёт себя также как и корреляция Пирсона, если пользователи оценили одинаковые объекты. Оценки разными пользователями разных объектов всё ещё выпадают из числителя, однако увеличивают знаменатель.

Вычисление предсказанных оценок

Идеи, на которых строятся способы расчёта оценок, заключаются в следующем:

- Чем более пользователи схожи между собой, тем большую роль играет вкус одного пользователя для другого, это можно смоделировать, взвесив оценки других пользователей.

- Пользователи взаимодействуют с объектами по-разному: кто-то ставит более низкие оценки, кто-то оценивает только понравившиеся объекты и т. д. Это значит, что оценки разных пользователей описываются

разными распределениями, что можно учесть с помощью стандартизированной оценки [4].

Таким образом, оценки вычисляются по формуле:

$$r_{ui} = \bar{r}_u + \sigma_u \frac{\sum_{u' \in U_u} sim(u, u') \frac{(r_{u'i} - \bar{r}_{u'})}{\sigma_{u'}}}{\sum_{u' \in U_u} sim(u, u')},$$

где U_u — множество похожих на пользователя u пользователей

Итак, для составления матрицы предсказанных оценок необходимо выполнить следующие шаги:

1. Рассчитать коэффициенты схожести между всеми пользователями.
2. Последовательно выбирая подмножество схожих пользователей для конкретного пользователя, вычислить предсказанные оценки.

Выше описан user-based подход, но, с точностью до перестановки пользователей и объектов, всё вышесказанное верно и для item-based подхода.

Так как в конечном итоге алгоритм составляет матрицу оценок, есть возможность совместить два этих подхода:

$$r_{ui} = (1 - \alpha) UB(u, i) + \alpha IB(u, i)$$

Выбирая α, можно менять вклад каждого из подходов в конечную оценку.

1.2 Model-based

Эта техника предполагает использование алгоритмов разложения матриц и многослойных нейронных сетей. Задача заключается в том, чтобы выявить скрытые характеристики объектов для получения недоступной ранее информации и сокращения размерности.

1.2.1 Алгоритмы разложения матриц в контексте рекомендательных систем

Как правило, пользователи взаимодействуют с небольшим количеством объектов, вследствие чего матрицы пользовательских оценок обычно сильно

разряжены. Это отрицательно сказывается на производительности рекомендательных систем. Идея, лежащая в основе применения матричного разложения в рекомендательных системах, заключается в том, что характеристики или предпочтения пользователя могут определяться небольшим количеством скрытых факторов, которые называются эмбеддингами.

Суть матричного разложения представлена на рисунке 1:

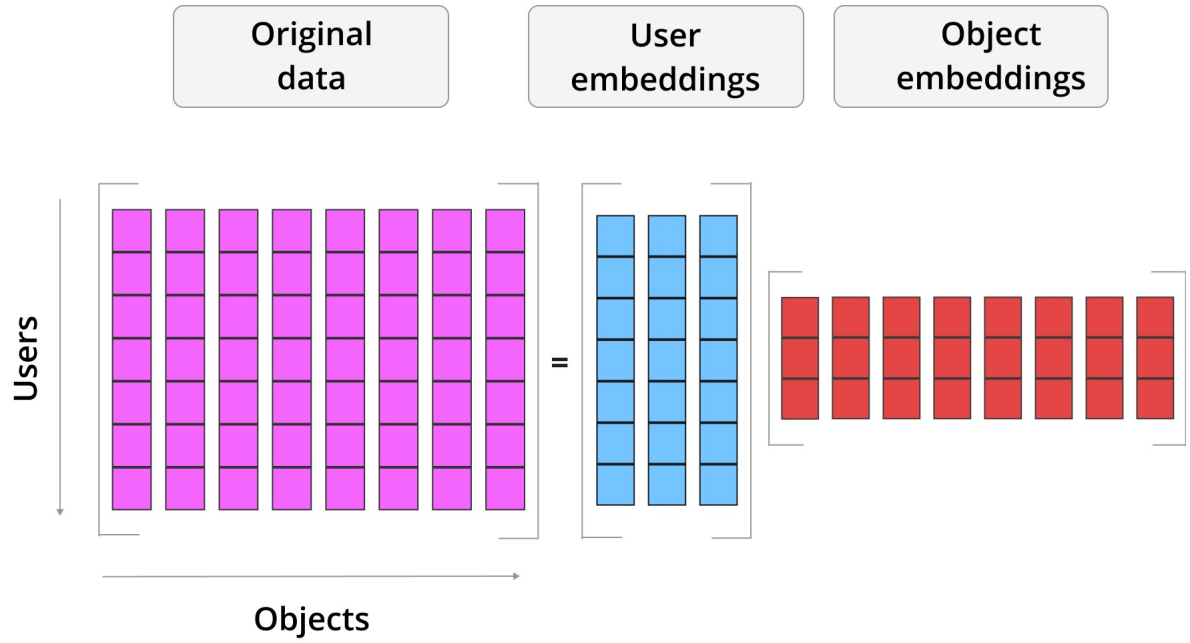

Рис. 1

Эмбеддинги можно понимать как вектора скрытых свойств, присущих пользователям и объектам, как правило, обладающие низкой размерностью. Матрицы, полученные в ходе разложения, являются матрицами эмбеддингов.

Задача разложения матрицы может быть переформулирована как задача оптимизации с определённой функцией потерь. Для нахождения разложения матрицы эмбеддингов инициализируются случайными элементами, а далее, путем минимизации ошибки рассчитываются актуальные значения. В процессе матричного разложения значения в исходной матрице аппроксимируются значениями реконструированной матрицы, разряженность которой снижается, что значит, что мы получаем значения для неизвестных (нулевых) элементов исходной матрицы. Предсказанная оценка для пары пользователь-объект суть произведение соответствующих векторов матриц разложения.

Alternating Least Squares

Зададим предсказанную оценку как произведение векторов, соответствующих пользователю и объекту:

$$\hat{r_{ui}} = U_u \cdot V_i$$

где U_u, V_i — вектора скрытых признаков пользователя и объекта соответственно.

Тогда функция потерь может быть задана следующим образом:

$$L = \sum_{u,i \in S} (r_{ui} - U_u \cdot V_i)^2$$

где S — множество пар пользователей/объектов, с которыми производилось взаимодействие.

Для защиты от переобучения используется L2-регуляризация [6]

$$L = \sum_{u,i \in S} (r_{ui} - U_u \cdot V_i)^2 + \lambda_U \sum_u \|U_u\|^2 + \lambda_V \sum_i \|V_i\|^2$$

где λ_U, λ_V — гиперпараметры модели для регуляризации.

Метод предполагает минимизацию ошибки путём последовательного фиксирования одного набора векторов скрытых признаков и обновления другого. Дифференцируя отдельно по признакам пользователей, отдельно по признакам объектов, получают выражения для производной:

$$\frac{\partial L}{\partial U_u} = -2 \sum_i (r_{ui} - U_u \cdot V_i) V_i + 2\lambda_U U_u$$

$$\frac{\partial L}{\partial V_i} = -2 \sum_u (r_{ui} - U_u \cdot V_i) U_i + 2\lambda_V V_i$$

Приравнивая производную к нулю, получают выражения для искомых векторов:

$$U_u = r_u V (V^\intercal V + \lambda_U E)^{-1}$$

$$V_i = r_i U (U^\intercal U + \lambda_V E)^{-1}$$

Выбрав размерность эмбеддингов и параметры регуляризации, итеративно рассчитывая новые значения целиком сначала для одного набора векторов, а затем для другого, вычисляют матрицы разложений.

Stochastic Gradient Descent

Оценки разных пользователей описываются разными распределениями, то же верно и для объектов взаимодействия. Чтобы учесть это, в модель добавляют поправку на центр распределения.

Тогда предсказанная оценка представляется следующим образом:

$$\hat{r_{ui}} = \mu + \mu_u + \mu_i + U_u \cdot V_i$$

где U_u, V_i — вектора скрытых признаков пользователя и объекта соответственно, μ, μ_u, μ_i — общее среднее, среднее пользователя и среднее объекта соответственно.

Функция потерь с L2-регуляризацией:

$$L = \sum_{u,i \in S} (r_{ui} - \hat{r_{ui}}))^2 + \lambda_U \sum_u (\|U_u\|^2 + \|\mu_u\|^2) + \lambda_V \sum_i (\|V_i\|^2 + \|\mu_i\|^2)$$

где S — множество пар пользователей/объектов, с которыми производилось взаимодействие.

Минимизация производится посредством стохастического градиентного спуска. Значения градиентов по переменным:

$$\frac{\partial L}{\partial U_u} = -2 \sum_i (r_{ui} - \hat{r_{ui}}) V_i + 2\lambda_U U_u$$

$$\frac{\partial L}{\partial V_i} = -2 \sum_u (r_{ui} - \hat{r_{ui}}) U_i + 2\lambda_V V_i$$

$$\frac{\partial L}{\partial \mu_u} = -2 \sum_i (r_{ui} - \hat{r_{ui}}) + 2\lambda_U \mu_u$$

$$\frac{\partial L}{\partial \mu_i} = -2 \sum_u (r_{ui} - \hat{r_{ui}}) + 2\lambda_V \mu_i$$

На каждой итерации вектора обновляются на основе градиента ошибки, который рассчитывается для одной или, если обучение ведётся батчами, нескольких пар оценок. Таким образом уравнения для обновления векторов признаков записываются как:

$$U_u = U_i + \nu((r_{ui} - \hat{r_{ui}})V_i - \lambda_U U_i)$$

$$V_i = V_i + \nu((r_{ui} - \hat{r_{ui}})U_u - \lambda_V V_i)$$

$$\mu_u = \mu_u + \nu(r_{ui} - \hat{r_{ui}} - \lambda_U \mu_u)$$

$$\mu_i = \mu_i + \nu(r_{ui} - \hat{r_{ui}} - \lambda_V \mu_i)$$

где ν — коэффициент скорости обучения.

Выбрав размерность эмбеддингов, параметры регуляризации и скорость обучения, итеративно вычисляют матрицы разложений.

Probabilistic Matrix Factorization

Следующий подход был предложен в [7]. Идея строится на Байесовском выводе. Предполагается, что множество оценок нормально распределено относительно предсказанных оценок с общей дисперсией:

$$p(R|U, V, \sigma^2) = \prod_{u,i \in S} N(r_{ui}|U_u V_i, \sigma^2) \tag{2}$$

где S — множество пар пользователей/объектов, с которыми производилось взаимодействие, N — нормальное распределение.
Предполагается, что:

- Оценки независимы
- Оценки нормально распределены с дисперсией σ^2

Распределения векторов признаков задаётся распределением Гаусса с центром в нуле:

$$\begin{aligned} p(U|\sigma_U^2) &= \prod_u N(U_u|0, \sigma_U^2) \\ p(V|\sigma_V^2) &= \prod_i N(V_i|0, \sigma_V^2) \end{aligned} \tag{3}$$

Предполагается, что:

- Вектора пользователей и объектов независимы между собой
- Вектора пользователей и объектов нормально распределены с дисперсией σ^2

Задав априорные распределения, можно перейти к апостериорному выводу. Исходя из правила Байеса:

$$p(U, V|R, \sigma^2) = \frac{p(R|U, V, \sigma^2)p(U, V|\sigma^2)}{p(R|\sigma^2)} \propto p(R|U, V, \sigma^2)p(U, V|\sigma^2)$$

Так как вектора признаков независимы, можно переписать выражение:

$$p(U, V|R, \sigma^2) = p(R|U, V, \sigma^2)p(U|\sigma_U^2)p(V|\sigma_V^2)$$

С учётом (2) и (3):

$$p(U, V|R, \sigma^2) = \prod_{u,i \in S} N(r_{ui}|U_u V_i, \sigma^2) \prod_u N(U_u|0, \sigma_U^2) \prod_i N(V_i|0, \sigma_V^2)$$

Максимизация этой функции эквивалентна максимизации логарифмированного её варианта.

$$ln\ p(U, V|R, \sigma^2) = -\frac{1}{2\sigma^2} \sum_{u,i \in S} (r_{ui} - U_u V_i)^2 - \frac{1}{2\sigma_U^2} \sum_u \|U_u\|^2 - \frac{1}{2\sigma_V^2} \sum_i \|V_i\|^2$$

Таким образом функция потерь представляется в знакомом виде:

$$L = \frac{1}{2}(\sum_{u,i \in S} (r_{ui} - U_u V_i)^2 - \lambda_U \sum_u \|U_u\|^2 - \lambda_V \sum_i \|V_i\|^2)$$

где $\lambda_U = \frac{\sigma^2}{\sigma_U^2}$, $\lambda_V = \frac{\sigma^2}{\sigma_V^2}$.

Особенность PMF заключается в возможности пересчёта параметров регуляризации на каждой итерации, что позволяет вносить новую информацию в модель по мере обучения. Для вычисления матриц разложения можно применить ALS или SGD подходы, рассмотренные ранее.

1.2.2 Нейронные сети для рекомендаций

Подход с использованием нейронных сетей можно рассматривать как модификацию подхода с использованием матричного разложения. Идея для обучения нейронной сети основывается на том факте, что матрицы эмбеддингов, получаемые в ходе разложения матриц, могут быть смоделированы нейросетью. Для этого используется один из слоёв сети, в качестве параметров которого выступают случайным образом инициализированные эмбеддинги. Параметры этого слоя передаются на вход последующим слоям нейросети и в процессе обучения изменяются таким образом, чтобы давать корректные значения оценок для пар пользователей и объектов. На рисунке 2 приведена схема использования нейросетей для решения описанной задачи.

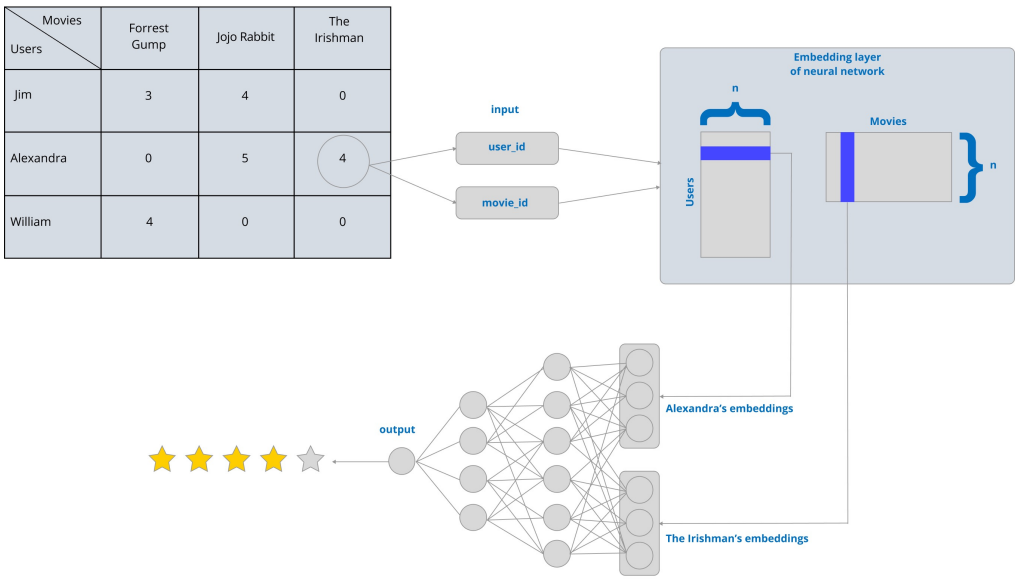

Рис. 2

Архитектура нейросети представлена на рисунке 3

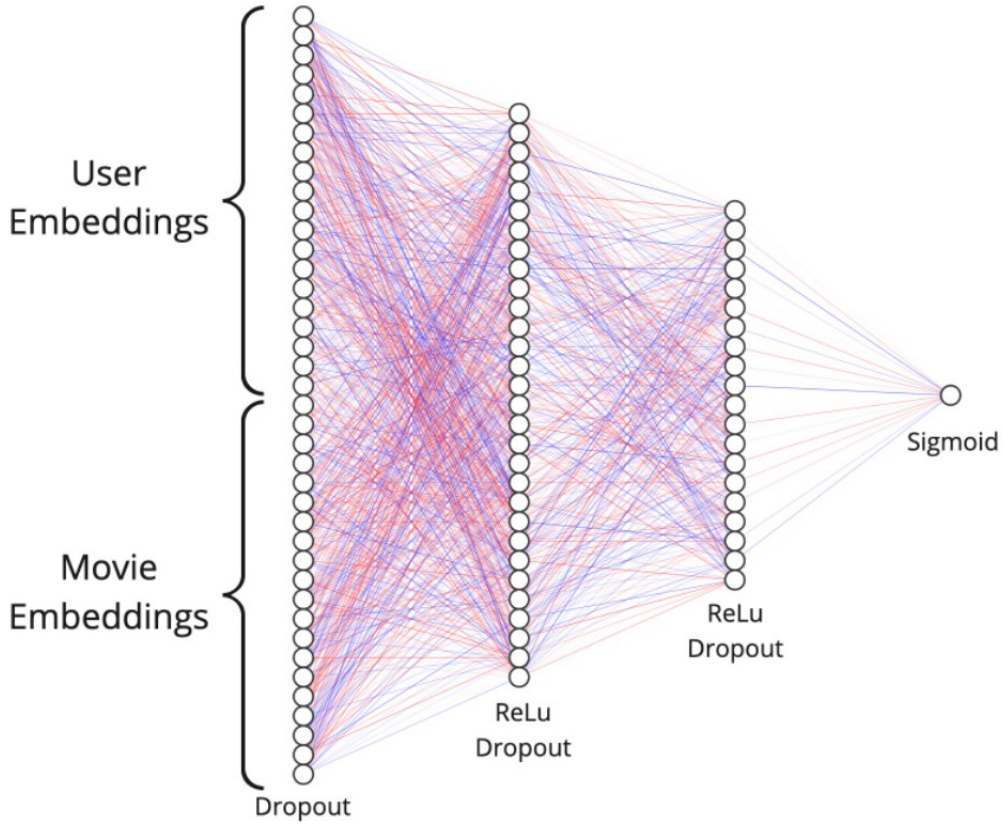

Рис. 3

§2 Рекомендательные системы на основе содержимого

Техники коллаборативной фильтрации обладают существенным минусом — для них актуальна проблема "холодного старта". У новых объектов отсутствует история взаимодействий, в следствие чего модель не рекомендует их пользователям, и не происходит накопление информации. Как правило, у объектов, с которыми взаимодействуют пользователи, можно заранее выделить некоторые признаки. Это может быть год выхода, режиссёр, актёрский состав фильма; цена или категория товара. На основе подобных признаков работают **content-based** алгоритмы.

Машинные алгоритмы не могут напрямую работать с нечисловыми признаками, такими как тэги фильма. Чтобы получить числовое представление таких признаков, можно использовать **TF-IDF** [8] (TF — term frequency, IDF — inverse document frequency). Объекты представляются в виде документов, в которые включаются все словесные признаки. TF-IDF мера вычисляется по следующим формулам:

TF — отношение числа вхождений некоторого слова к общему числу слов документа.

$$tf(t,d) = \frac{n_t}{\sum_k n_k}$$

где n_k — число вхождений слова k в документ d.

IDF — инвертированная частота, с которым слово встречается в корпусе.

$$idf(t,D) = log\ \frac{\|D\|}{\|\{d_i \in D \| t \in d_i\}\|}$$

где D — множество документов в корпусе.

Мера **TF-IDF**.

$$tf\text{-}idf(t,d,D) = tf(t,d) \times idf(t,D)$$

Но такое представление фрагментированно и избыточно, так как каждому слову будет соответствовать отдельное измерение, и вектора, соответствующие синонимичным словам, будут ортогональны. Чтобы сжать пространство можно использовать автоэнкодеры [9] - специальные нейросети, обученные предсказывать то же, что получили на вход. Архитектура автоэнкодера пред-

ставлена на рисунке 4:

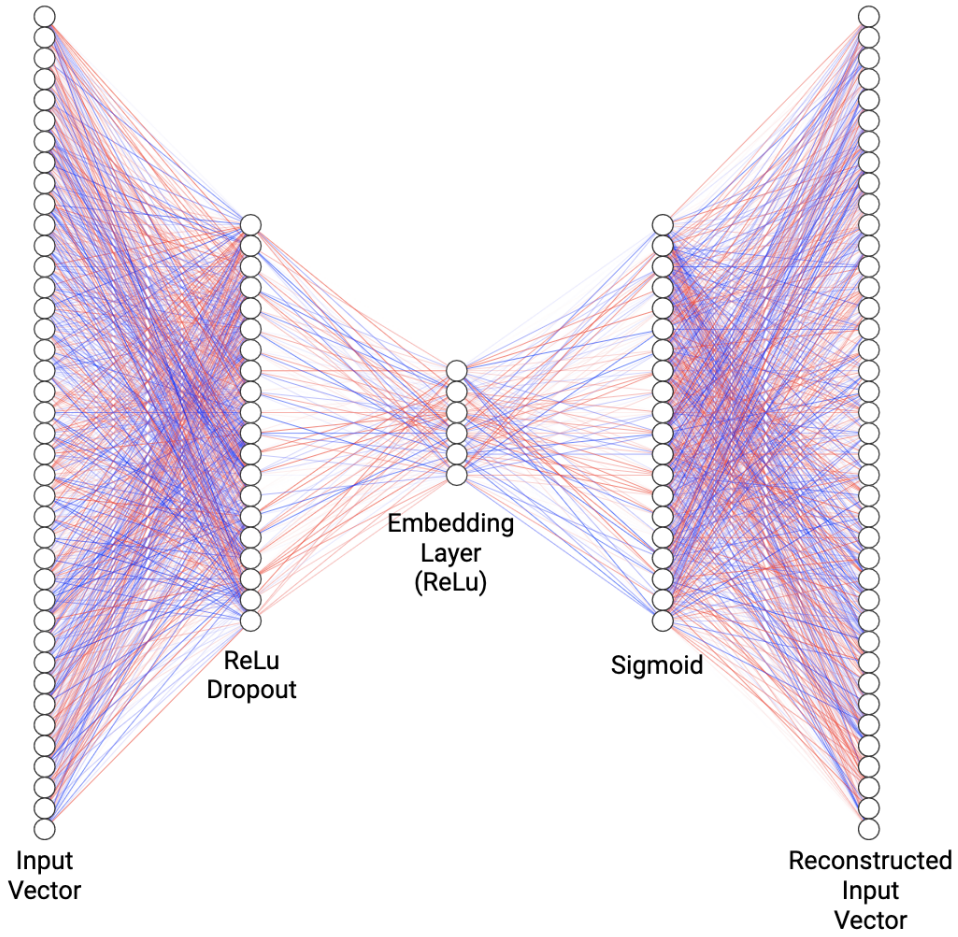

Рис. 4

Данная архитектура сжимает многомерное TF-IDF пространство в n-мерное. Первая половина сети (энкодер) кодирует вектор, соответствующий объекту, а вторая половина (декодер) реконструирует оригинальные данные. Сжатое представление в n-мерном пространстве моделируется центральным слоем сети.

Полученные таким образом векторные представления объектов дают возможность включать в рекомендации объекты, не имеющие истории взаимодействия. Мы можем дополнительно получить вектор пользователя в пространстве, порождённым автоэнкодером, взяв взвешенную сумму векторов

объектов, с которыми тот уже взаимодействовал.

$$U_i^c = \sum_{j \in S_i} R_{ij}\ I_j^c$$

где S_i — множество объектов, с которыми взаимодействовал пользователь i, R — матрица оценок, U^c, I^c — матрицы эмбеддингов пользователей и объектов.

Тогда релевантность объекта j для пользователя i может быть вычислена с помощью косинусного сходства:

$$rel_{ij} = 0.5\ \frac{U_i^c \cdot I_j^c}{\|U_i^c\| \|I_j^c\|} + 0.5,$$
$$rel_{ij} \in [0, 1]$$

Соответственно, предсказанная оценка:

$$\hat{R_{ij}} = r_{min} + (r_{max} - r_{min}) rel_{ij}$$

где r_{min}, r_{max} — минимально и максимально возможные оценки.

Кроме того, есть возможность рекомендовать похожие между собой объекты, вычисляя похожесть через косинусное сходство.

§3 Гибридные рекомендательные системы

Гибридная модель сочетает в себе коллаборативные модели с контентными. Гибридная рекомендательная система решает 3 задачи:

1. Рекомендация пользователям релевантных объектов.
2. Рекомендация объектов, похожих на заинтересовавший пользователя объект с учётом предпочтений пользователя.
3. Рекомендация объектов, похожих на заинтересовавший пользователя объект без учёта предпочтений пользователя.

Гибридный подход предполагает двухэтапное ранжирование.

Задача 1:

Сначала коллаборативная модель решает какие n объектов являются релевантными по отношению к пользовательскому запросу. Затем контентная модель самостоятельно задаёт порядок ранжирования или вносит в него поправки, используя свои знания о релевантности объектов.

Задача 2:

В случае рекомендации похожих объектов с учётом пользовательских предпочтений происходит наоборот — контентная модель отбирает n похожих объектов, а коллаборативная модель сортирует их с учётом весов, присвоенных объектам контентной моделью, или без.

Задача 3:

И коллаборативная, и контентная модели обладают векторными представлениями пользователей и объектов. Это позволяет каждой модели делать вывод о том, какие объекты являются похожими, используя косинусную дистанцию.

Ранжирование с учётом весов присвоенных обеими моделями математически выражается так:

$$rel_{ij} = (1 - \alpha)M_1(i, j) + \alpha M_2(i, j)$$

Выбирая α, можно менять вклад модели в конечную оценку.

Глава 2. Программная реализация

§1 Выбор инструментов разработки

Рекомендательные алгоритмы реализовывались на языке Python[12]. Выбор обусловлен наличием большого количества библиотек для работы с математическими вычислениями, с алгоритмами машинного обучения и нейросетями, а также лёгкость прототипирования на нём. Основные используемые библиотеки:

- **NumPy**[13] – для работы с матричными вычислениями
- **pandas**[14] – для работы с большими объёмами данных, в том числе для препроцессинга
- **scikit-learn**[15] – для организации обучения моделей машинного обучения: разбиение данных на обучающую и тестовую выборки, вычисление метрик
- **Matplotlib**[17] – для построения графиков
- **TensorFlow**[16] – для обучения нейросетевых моделей

Python – интерпретируемый язык. Эта категория языков, как правило, медленнее компилируемых. Однако, библиотеки, реализующие сложные математические вычисления, написаны на компилируемых языках, таких как C[18] или C++[19]. Это делает код, использующий эти библиотеки, очень быстрым. Таким образом, в данном контексте выбор Python в качестве языка программирования упрощает разработку, сохраняя при этом производительность программ.

§2 Структура проекта

Каждый рекомендательный алгоритм представляет из себя отдельную модель (класс). Все модели наследуются от базового класса, в котором имплементированы большое количество рутины, возникающей в ходе обучения:

- Препроцессинг данных
- Инициализация модели
- Разбиение данных на обучающую и тестовую выборки
- Построение графиков и мониторинг процесса обучения
- Сохранение параметров обучения модели и её выхода
- Оценка качества модели
- Поиск гиперпараметров (таких как размерность векторных представлений или количество эпох обучения), дающих наилучшее качество

Каждый класс содержит в себе программную реализацию конкретного алгоритма. В отдельных пакетах реализуются загрузка данных и метрики оценивания.

Процесс обучения модели в общем виде представлен алгоритмом 1:

Algorithm 1: Процесс обучения

Result: обученная модель
инициализировать модель;
while *эпоха обучения != максимум эпох* **do**
    сделать шаг алгоритма;
    вычислить метрики;
end
вычислить итоговое качество;

Самым важным на этапе исследования является подбор гиперпараметров модели, приводящих к оптимальному качеству. Поиск оптимальных гиперпараметров представлен алгоритмом 2:

Algorithm 2: Поиск оптимальных гиперпараметров

Result: наилучший набор гиперпараметров

foreach *набор гиперпараметров* **do**

    инициализировать модель;
    обучить модель;
    вычислить итоговое качество;

end

сравнить качество разных наборов гиперпараметров;

Глава 3. Эффективность рекомендаций

§1 Метрики оценивания

1.1 Регресионные метрики

Точность рекомендательных систем обычно рассчитывается по одной из двух метрик: среднеквадратратическая ошибка (RMSE) и средняя абсолютная ошибка (MAE). Обе этих метрики хорошо подходят для этой задачи, так как легко интерпретируются. Однако, в некоторых случаях одной из них может отдаваться предпочтение в зависимости от данных.

Mean Absolute Error

Средняя абсолютная ошибка представляет собой сумму абсолютных разностей между прогнозами и фактическими значениями.

$$MAE = \frac{1}{|R|} \sum_{u \in U, i \in I} |r_{ui} - \hat{r_{ui}}|$$

Основная черта этой метрики состоит в том, что она не никак не реагирует на большую величину конкретных разностей, но даёт целостное представление о точности работы рекомендательной системы.

Root Mean Squared Error

Среднеквадратическая ошибка усиливает вклад больших ошибок, будучи чувствительной к плохим предсказаниям.

$$RMSE = \sqrt{\frac{1}{|R|} \sum_{u \in U, i \in I} (r_{ui} - \hat{r_{ui}})^2}$$

Также стоит заметить, что эта метрика по определению всегда не меньше MAE.

В [5] утверждается, что MAE лучше описывает ошибки, имеющие равномерное распределение, в то время как RMSE — ошибки, имеющие нормальное распределение.

1.2 Метрики классификации

На практике часто не стоит задача сделать все предсказания как можно ближе к действительности, но правильно выделить некоторое количество подходящих объектов. Для этого удобно использовать F-меру и площадь под кривыми ошибок.

F-мера

Обычно классы релевантных и нерелевантных объектов не равны между собой по количеству членов, поэтому на каждом из классов имеет смысл ввести свою метрику, а затем объединить их в агрегированный критерий качества. Для этого используются точность (precision) — доля положительно классифицированных объектов, являющихся релевантными, от всех положительно классифицированных объектов и полнота (recall) — доля положительно классифицированных объектов, являющихся релевантными, от всех релевантных объектов.

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

На таблице 1 представлена так называемая матрица ошибок (confusion matrix), демонстрирующая смысл приведённых обозначений:

	$y = 1$	$y = 0$
$\hat{y} = 1$	True Positive (TP)	False Positive (FP)
$\hat{y} = 0$	False Negative (FN)	True Negative (TN)

Таблица 1: Confusion Matrix

Полнота и точность агрегируются в F-меру с параметром β, определяющим вклад точности в оценку:

$$F_\beta = (1 + \beta^2)\frac{precision \cdot recall}{\beta^2 \cdot precision + recall}$$

AUC-ROC и AUC-PR

Вещественные алгоритмы используют пороговое значение для определения принадлежности объекта к какому-либо классу, однако привязка метрики к конкретному порогу не является оптимальным решением. Оценить работу алгоритма в общем позволяет площадь под кривой ошибок (Area Under Curve). ROC-кривая строится по координатам True Positive Rate (TPR) и False Positive Rate (FPR), соответствующих определённому значению порога.

$$TPR = \frac{TP}{TP + FN},$$

$$FPR = \frac{FP}{FP + TN},$$

где TPR — полнота, FPR — доля неверно классифицированных объектов класса нерелевантных объектов. PR-кривая аналогично строится по координатам precision и recall. Примеры кривых приведены ниже на рисунках 5 и 6:

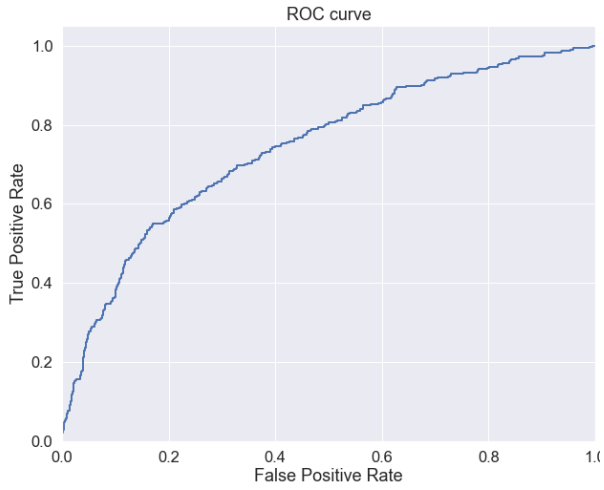

Рис. 5: График ROC-кривой

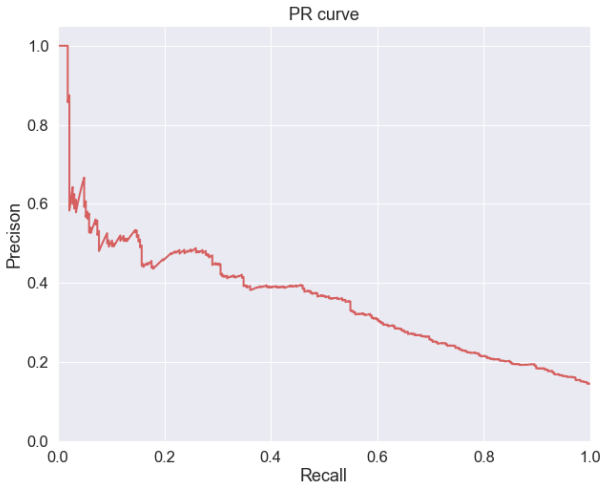

Рис. 6: График PR-кривой

Каждая точка на графике соответствует конкретному пороговому значению. Чем больше площадь под кривой, тем выше качество классификации.

1.3 Метрики качества ранжирования

Зачастую может быть недостаточно предсказать оценку, которую пользователь поставит объекту, или то, окажется ли объект полезным. В таком случае перед рекомендательной системой ставится задача ранжирования. Задача ранжирования заключается в том, чтобы отсортировать набор элементов исходя из их релевенатности по отношению к пользователю. В таком случае рекомендательный алгоритм помогают оценить метрики ранжирования.

Задача ранжирования может быть сформулирована следующим образом. Рассмотрим множество пользователей $U = \{u_i\}_{i=1}^{N}$ и множество объектов $V = \{v_j\}_{j=1}^{M}$. Результатом работы рекомендательного алгоритма является отображение $r_i : V \to \mathbb{R}$, присваивающее каждому объекту вес относительно конкретного пользователя. Конечный набор весов задаёт для каждого пользователя перестановку $\{n_1, \ldots, n_M\}$. Для оценки качества необходимо иметь знание о степени релевантности ранжируемых объектов $\hat{r}_i$ - отображение, задающее перестановку $\{\hat{n}_1, \ldots, \hat{n}_M\}$. Истинное знание о релевантности может быть получено как за счёт имеющихся данных, так и на основе экспертной оценки.

Cumulative Gain(CG)

CG представляет собой сумму истинных весов для p релевантных объектов:

$$CG_p = \sum_{j=1}^{p} \hat{r}_{n_j}$$

где $\hat{r}_{n_j}$ — истинная степень релевантности объекта на позиции n_j.

У метрики есть существенный недостаток - она нечувствительна к изменению порядка внутри p релевантных объектов.

Discounted Cumulative Gain (DCG)

DCG задаётся как [10]:

$$DCG_p = \sum_{j=1}^{p} \frac{\hat{r}_{n_j}}{\log_2(j+1)}$$

где $\hat{r}_{n_j}$ — истинная степень релевантности объекта на позиции n_j.

За счёт знаменателя метрика учитывает порядок объектов в списке. Выбор логарифма в качестве функции дисконтирования обусловлен тем, что порядок объектов в начале списка важнее, чем в конце. Теоретическое обоснование было дано в [11]. Не хватает только нормализации, чтобы давать сравнимые значения для пользователей с разным количество рекомендаций.

Normalized Discounted Cumulative Gain (NDCG)

Чтобы приобрести устойчивость к количеству элементов в списке - p, достаточно посчитать DCG для идеально отранжированного списка:

$$NDCG_p = \frac{DCG_p}{IDCG_p}$$

где IDCG - DCG для идеально отранжированного списка

$$IDCG_p = \sum_{j=1}^{p} \frac{\hat{r}_{\hat{n}_j}}{\log_2(j+1)}$$

Значение метрики всегда будет лежать в сегменте $[0, 1]$.

§2 Сравнение алгоритмов

Для сравнения алгоритмов используются данные из MovieLens 20M Dataset. Датасет содержит 20 миллионов оценок и 465 тысяч тегов, поставленных 138 тысячами пользователей 27 тысячам фильмов. Обучение производилось на 1 миллионе оценок, поставленных 6 тысячами пользователей 5 тысячам фильмов. Примеры исходных данных приведены на таблицах 2, 3.

userId	movieId	rating
438	5055	4.0
462	8910	3.0
474	3793	4.5
559	518	2.0

Таблица 2: Данные по оценкам

userId	movieId	tag
62	7153	fantasy
318	778	dark comedy
424	1625	plot twist
474	140	journalism

Таблица 3: Данные по тегам

В задаче классификации определяются два класса объектов. Вероятность принадлежности объекта j к положительному классу рассчитывается следующим образом:

$$p_{ij} = \begin{cases} 0.5 + 0.5\frac{\hat{r}_{ij} - \hat{r}_i^{mean}}{\hat{r}_i^{max} - \hat{r}_i^{mean}}, & \hat{r}_{ij} - \hat{r}_i^{mean} >= 0 \\ 0.5 - 0.5\frac{\hat{r}_{ij} - \hat{r}_i^{mean}}{\hat{r}_i^{min} - \hat{r}_i^{mean}}, & \hat{r}_{ij} - \hat{r}_i^{mean} < 0 \end{cases}$$

где $\hat{r}_{ij}$ — предсказанная оценка,
$\hat{r}_i^{mean}$, $\hat{r}_i^{min}$, $\hat{r}_i^{max}$ — средняя, минимальная и максимальная предсказанные оценки пользователя.

На тестовом множестве считается, что объект относится к положительному классу, если пользователь поставил ему оценку выше своей средней.

2.1 Memory-based

Схожесть будет рассчитываться с помощью центрированного косинусного сходства:

$$sim(u,v) = \frac{\sum_{i \in I_u \cup I_v} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in I_u \cup I_v} (r_{ui} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_u \cup I_v} (r_{vi} - \bar{r}_v)^2}},$$

Предсказанная оценка вычисляется с использованием взвешивания оценок похожих пользователей и стандартизированной оценки:

$$r_{ui} = \bar{r}_u + \sigma_u \frac{\sum_{u' \in U_u} sim(u, u') \frac{(r_{u'i} - \bar{r}_{u'})}{\sigma_{u'}}}{\sum_{u' \in U_u} sim(u, u')},$$

Совместим user-based и item-based подходы, определяя вклад каждого из них с помощью параметра α. К чистым user-based и item-based алгоритмам относятся значения $\alpha = 0$ и $\alpha = 1$ соответственно.

$$r_{ui} = (1 - \alpha)UB(u, i) + \alpha IB(u, i)$$

Оценка качества алгоритма представлена на таблице 4

α	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
RMSE	0.8847	0.8732	0.8635	0.8560	0.8504	0.8471	0.8458	0.8468	0.8498	0.8551	0.8624
MAE	0.6619	0.6540	0.6473	0.6420	0.6381	0.6356	0.6347	0.6353	0.6376	0.6415	0.6470
NDCG	0.9576	0.9583	0.9593	0.9601	0.9602	0.9607	0.9607	0.9603	0.9598	0.9592	0.9587
F1	0.7172	0.7197	0.7224	0.7241	0.7254	0.7253	0.7257	0.7259	0.7239	0.7213	0.7199
AUC-ROC	0.7079	0.7126	0.7158	0.7181	0.7196	0.7207	0.7211	0.7202	0.7185	0.7157	0.7122
AUC-PR	0.7307	0.7358	0.7388	0.7406	0.7421	0.7433	0.7436	0.7417	0.7388	0.7353	0.7310

Таблица 4: Значения метрик в зависимости от α

Графики кривых при $\alpha = 0.6$ представлены на рисунках 7 и 8:

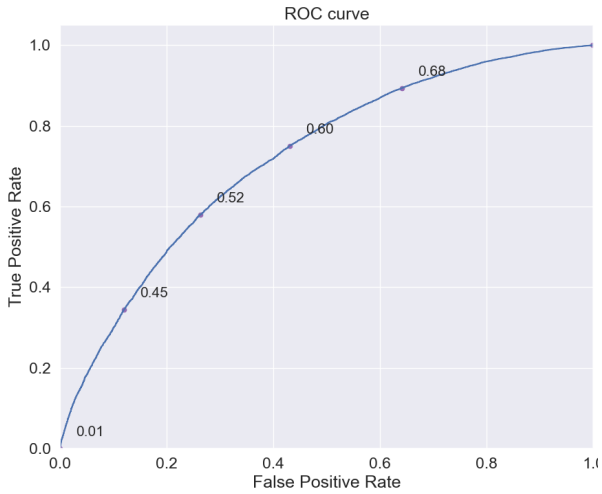

Рис. 7: График ROC-кривой

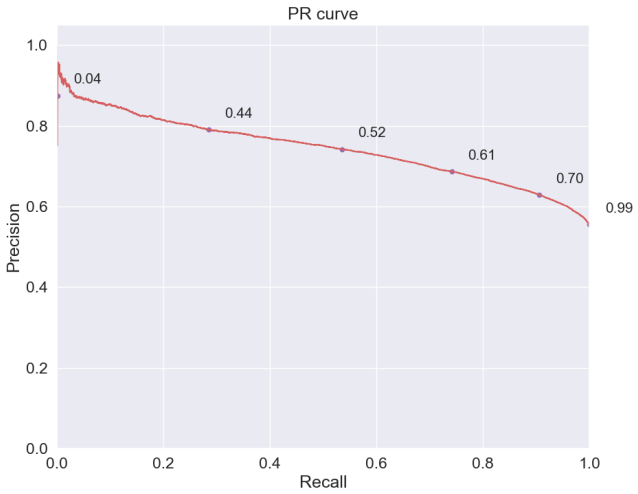

Рис. 8: График PR-кривой

2.2 Model-based

Alternating Least Squares

Оценка качества представлена на таблице 5

RMSE	MAE	NDCG	F1	AUC-ROC	AUC-PR
0.8754	0.6802	0.9609	0.7296	0.7040	0.7268

Таблица 5: Значения метрик для ALS

Графики ROC/PR-кривых приведены на рисунках 9, 10:

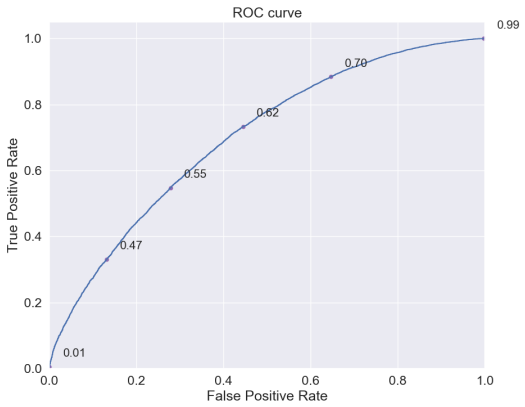

Рис. 9: График ROC-кривой для ALS

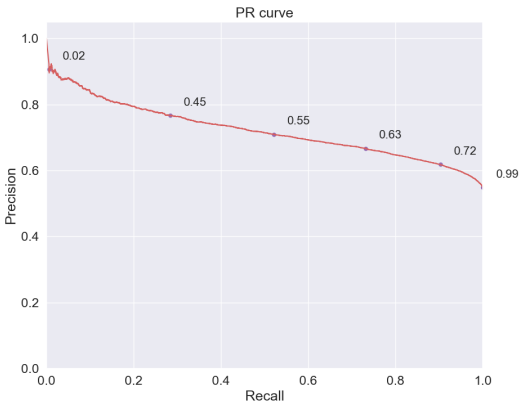

Рис. 10: График PR-кривой для ALS

Значения метрик в процессе обучения представлены на рисунках 11, 12

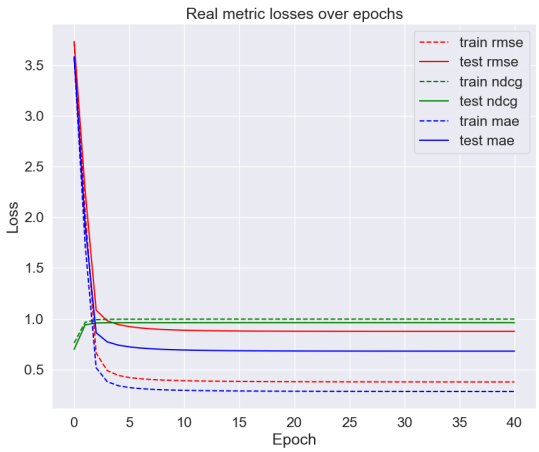

Рис. 11: Регрессионные метрики ALS

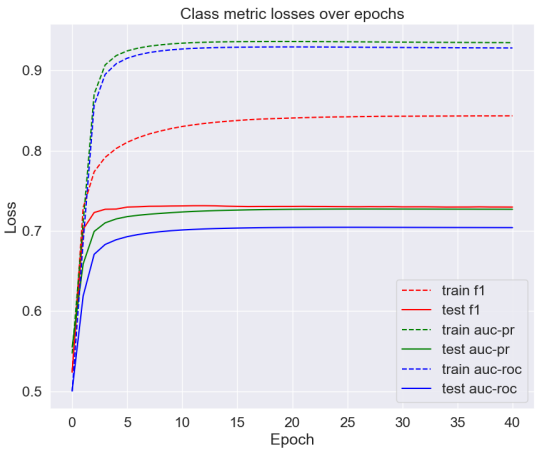

Рис. 12: Метрики классификации ALS

Stochastic Gradient Descent

Оценка качества представлена на таблице 6

RMSE	MAE	NDCG	F1	AUC-ROC	AUC-PR
0.8357	0.6373	0.9616	0.7287	0.7221	0.7428

Таблица 6: Значения метрик для SGD

Графики ROC/PR-кривых приведены на рисунках 13, 14:

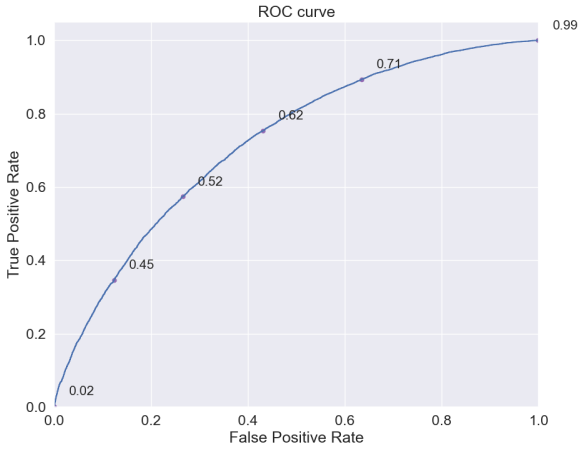

Рис. 13: График ROC-кривой для SGD

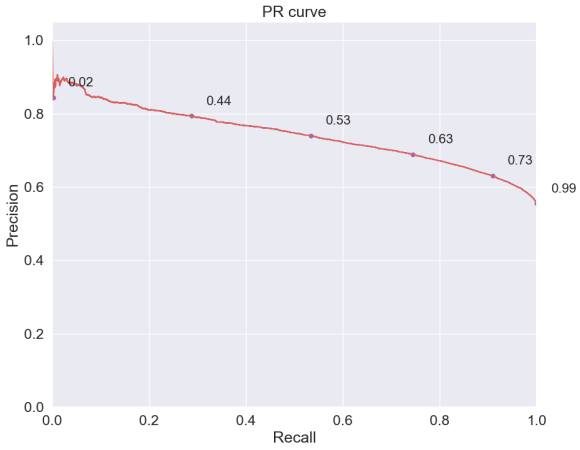

Рис. 14: График PR-кривой для SGD

Значения метрик в процессе обучения представлены на рисунках 15, 16

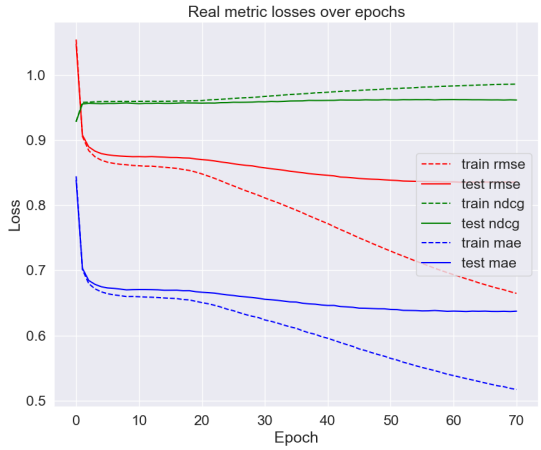

Рис. 15: Регрессионные метрики SGD

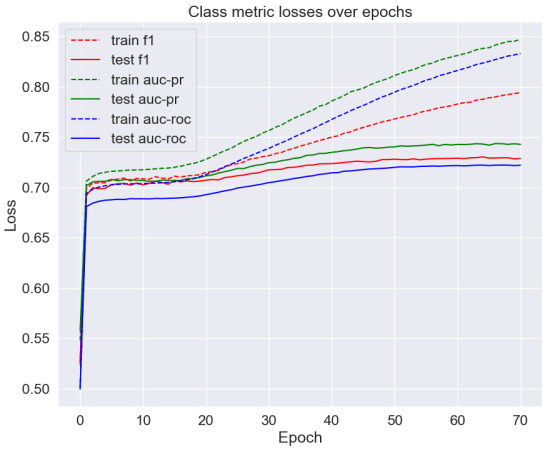

Рис. 16: Метрики классификации SGD

Probabilistic Matrix Factorization

Оптимизация функции потерь производилась посредством SGD. Оценка качества представлена на таблице 7

RMSE	MAE	NDCG	F1	AUC-ROC	AUC-PR
0.9135	0.7082	0.9554	0.7127	0.6780	0.7017

Таблица 7: Значения метрик для PMF

Графики ROC/PR-кривых приведены на рисунках 17, 18:

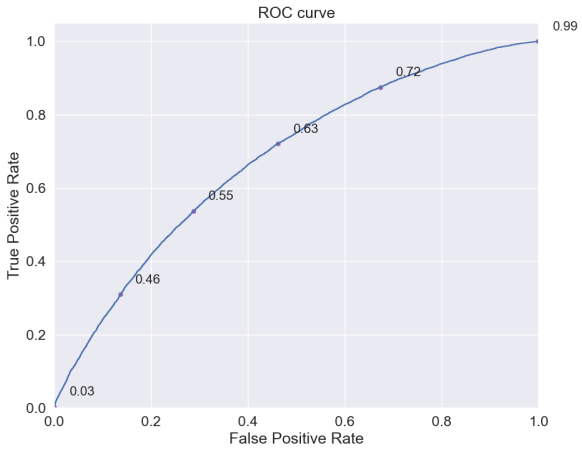

Рис. 17: График ROC-кривой для PMF

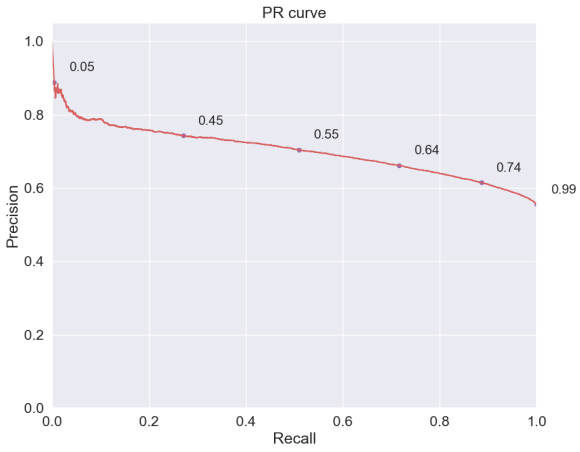

Рис. 18: График PR-кривой для PMF

Значения метрик в процессе обучения представлены на рисунках 19, 20

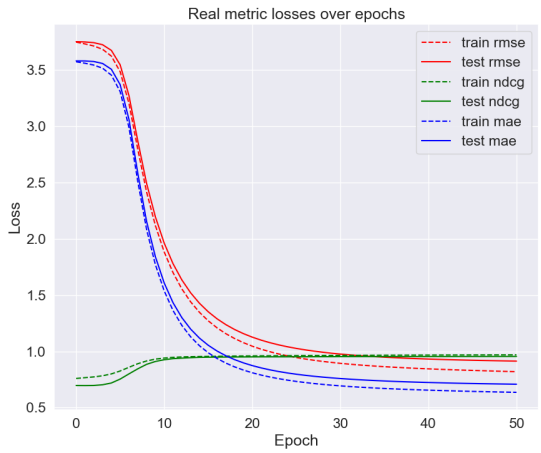

Рис. 19: Регрессионные метрики PMF

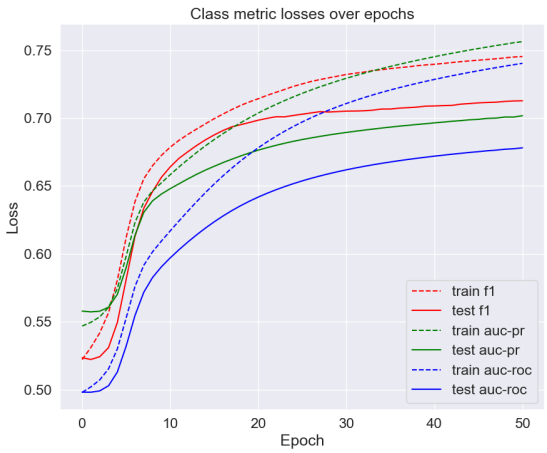

Рис. 20: Метрики классификации PMF

Neural Network

Оценка качества представлена на таблице 8

RMSE	MAE	NDCG	F1	AUC-ROC	AUC-PR
0.8587	0.6532	0.9581	0.7130	0.7010	0.7227

Таблица 8: Значения метрик для нейросети

Графики ROC/PR-кривых приведены на рисунках 21, 22:

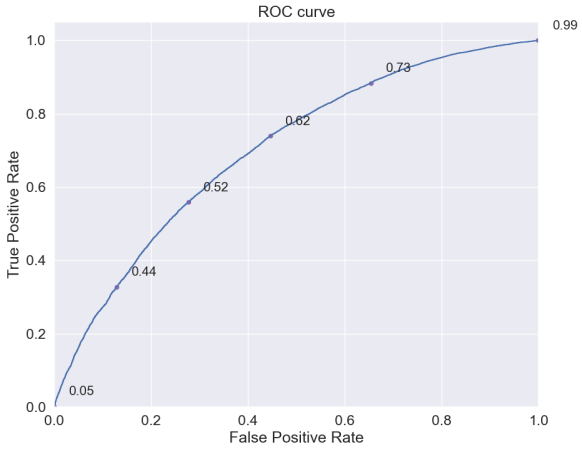

Рис. 21: График ROC-кривой для нейросети

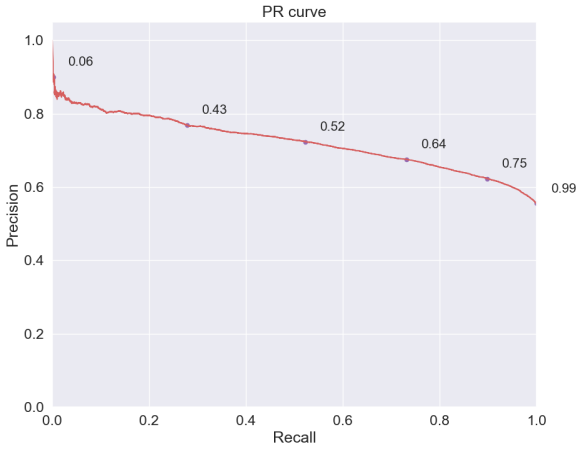

Рис. 22: График PR-кривой для нейросети

Значения метрик в процессе обучения представлены на рисунках 23, 24

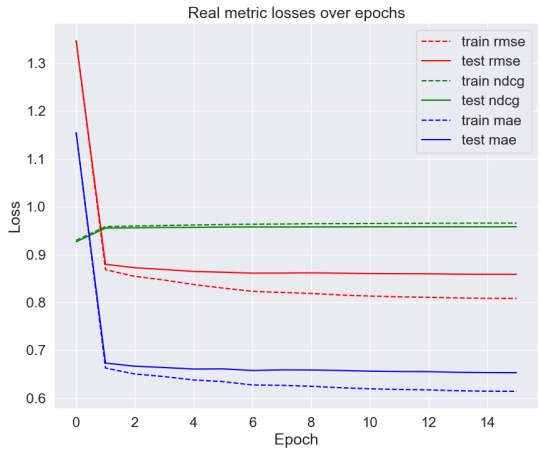

Рис. 23: Регрессионные метрики нейросети

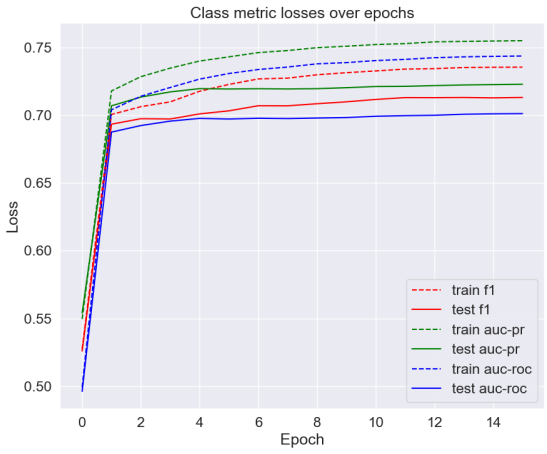

Рис. 24: Метрики классификации нейросети

2.3 Content-based

Оценка качества представлена на таблице 9

RMSE	MAE	NDCG	F1	AUC-ROC	AUC-PR
1.18735	0.8976	0.9449	0.6693	0.6037	0.6417

Таблица 9: Значения метрик

Графики ROC/PR-кривых приведены на рисунках 25, 26:

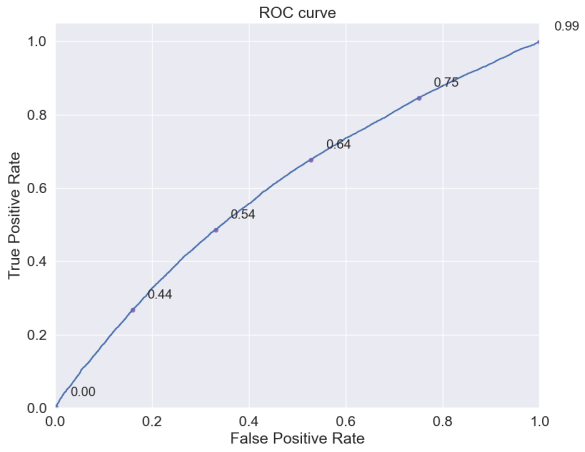

Рис. 25: График ROC-кривой

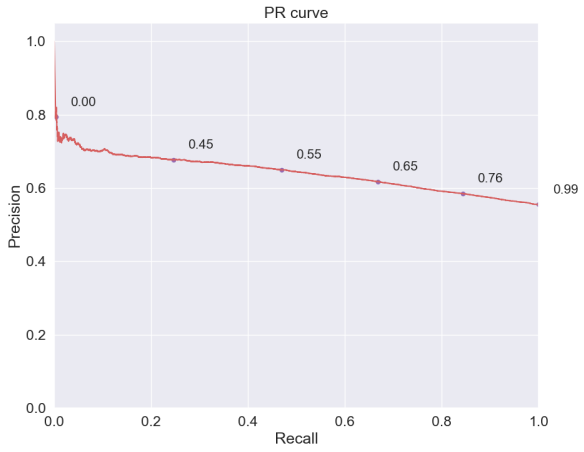

Рис. 26: График PR-кривой

Среднеквадратическая ошибка автоэнкодера в процессе обучения представлена на рисунке 27

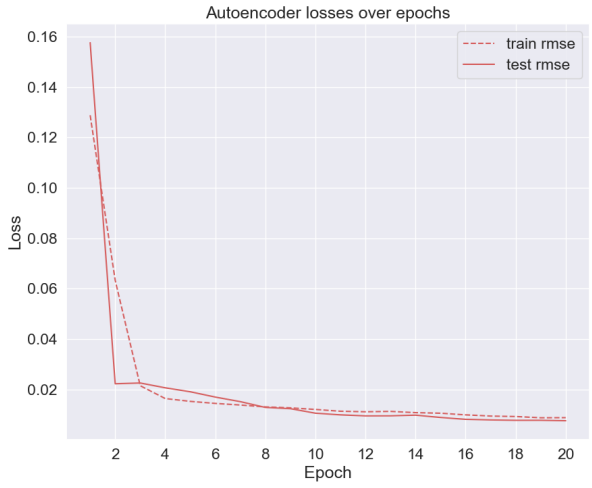

Рис. 27: Среднеквадратическая ошибка автоэнкодера

2.4 Hybrid

Для оценки гибридной модели в качестве коллаборативной модели использовалось SGD-разложение, а в качестве контентной - автоэнкодер, обученный сжимать tf-idf пространство тэгов.

Вклад каждой модели варьироваался посредством параметра α. К ответам чистых коллаборативной и контентной моделей относятся значения $\alpha = 0$ и $\alpha = 1$ соответственно.

$$\hat{r}_{ij} = (1 - \alpha)SGD(i, j) + \alpha C(i, j)$$

Оценка качества алгоритма представлена на таблице 10

α	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
RMSE	0.8357	0.8405	0.8535	0.8744	0.9026	0.9374	0.9782	1.0242	1.0748	1.1294	1.1873
MAE	0.6373	0.6383	0.6461	0.6599	0.6792	0.7040	0.7339	0.7685	0.8077	0.8510	0.8976
NDCG	0.9616	0.9619	0.9623	0.9625	0.9620	0.9613	0.9596	0.9581	0.9550	0.9503	0.9449
F1	0.7287	0.7293	0.7312	0.7318	0.7315	0.7309	0.7268	0.7199	0.7093	0.6914	0.6693
AUC-ROC	0.7221	0.7233	0.7241	0.7242	0.7228	0.7190	0.7116	0.6983	0.6765	0.6443	0.6037
AUC-PR	0.7428	0.7439	0.7446	0.7446	0.7428	0.7384	0.7309	0.7192	0.7017	0.6754	0.6417

Таблица 10: Значения метрик в зависимости от α

Графики кривых при $\alpha = 0.3$ представлены на рисунках 28 и 29:

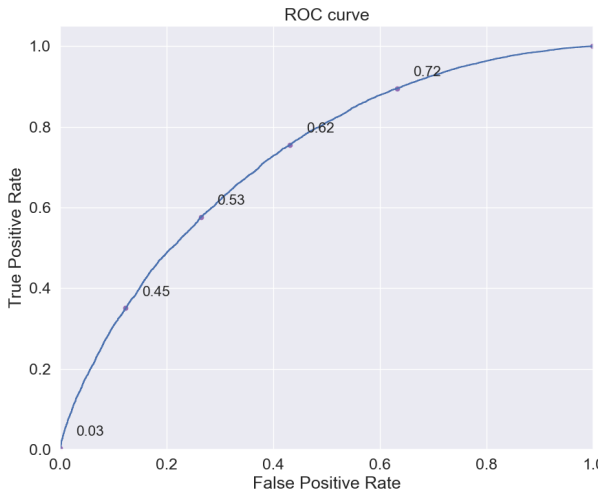

Рис. 28: График ROC-кривой

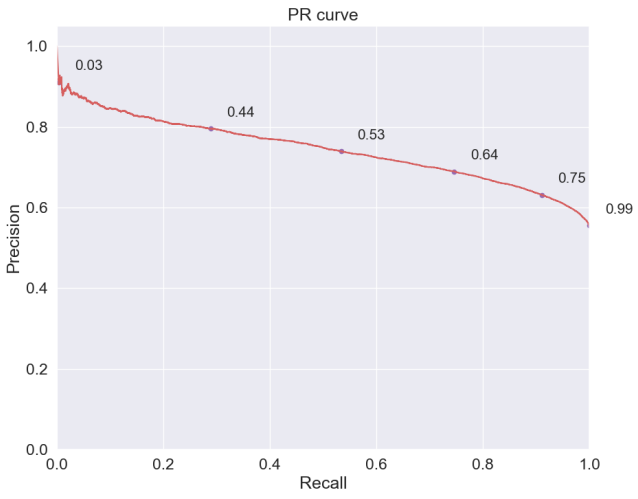

Рис. 29: График PR-кривой

Список литературы

[1] Resnick P., Iacovou N., Suchak M., Bergstrom P., Riedl J. «GroupLens: an open architecture for collaborative filtering of netnews». Proceedings of the 1994 ACM conference on Computer supported cooperative work, 1994, pp. 175-186.

[2] Herlocker J., Konstan J., Borchers A., Riedl J. «An algorithmic framework for performing collaborative filtering». Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, 1999, pp. 230-237.

[3] Breese J., Heckerman D., Kadie C. «Empirical Analysis of Predictive Algorithms for Collaborative Filtering». Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence, 1998, pp. 43-52.

[4] Herlocker J., Webster J., Jung S., Dragunov A., Holt T., Culter T., Haerer S. «A framework for collaborative information environments and unified access to distributed digital content». Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries, 2002. pp. 378-378.

[5] Chai T. and Draxler R. «Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avoiding RMSE in the literature». Geosci. Model Dev., 7, 2014, pp. 1247–1250.

[6] Andrew Ng «Feature selection, L1 vs. L2 regularization, and rotational invariance». Proceedings of the twenty-first international conference on Machine learning, 2004, p. 78.

[7] Mnih A. and Salakhutdinov R. «Probabilistic matrix factorization ». Proceedings of the 20th International Conference on Neural Information Processing Systems, 2007, pp. 1257-1264.

[8] Roelleke T. and Wang J. «TF-IDF Uncovered: A Study of Theories and Probabilities ». Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2008, pp. 435-442.

[9] Baldi, Pierre «Autoencoders, Unsupervised Learning and Deep Architectures ». Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning Workshop - Volume 27, 2011, pp. 37-50.

[10] Järvelin K., Kekäläinen J. «Cumulated gain-based evaluation of IR techniques ». ACM Transactions on Information Systems 20(4), 2002, pp. 422–446.

[11] Wang Y. Wang L., Li Y., Di H., Tie-Yan L., Wei C. «A Theoretical Analysis of NDCG Type Ranking Measures ». Proceedings of the 26th Annual Conference on Learning Theory - Volume 30, 2013.

[12] Python – an open source programming language [Электронный ресурс]. — URL: https://www.python.org/

[13] NumPy – an open source project aiming to enable numerical computing with Python [Электронный ресурс]. — URL: https://numpy.org/

[14] pandas – an open source data analysis and manipulation tool [Электронный ресурс]. — URL: https://pandas.pydata.org/

[15] scikit-learn – free software machine learning library [Электронный ресурс]. — URL: https://scikit-learn.org/

[16] TensorFlow – an end-to-end open source platform for machine learning [Электронный ресурс]. — URL: https://www.tensorflow.org/

[17] Matplotlib – a comprehensive library for creating static, animated, and interactive visualizations [Электронный ресурс]. — URL: https://matplotlib.org/

[18] C – a general-purpose, procedural computer programming language [Электронный ресурс]. — URL: https://www.iso.org/standard/74528.html/

[19] C++ – a general-purpose programming language [Электронный ресурс]. — URL: https://isocpp.org/