

## **Разработка системы рекомендаций на основе коллаборативной фильтрации с использованием нейронной сети**

*Эрдман Александр Алексеевич*

*Приамурский государственный университет имени Шолом-Алейхема*

*Студент*

### **Аннотация**

В статье рассматривается разработка системы рекомендаций на основе коллаборативной фильтрации с использованием нейронной сети. Для реализации используются библиотеки Python и TensorFlow в среде Google Colaboratory. Результатом исследования является модель, способная предсказывать рейтинги фильмов для пользователей и генерировать персонализированные рекомендации.

**Ключевые слова:** TensorFlow, Python, нейронные сети

## **Development of a recommendation system based on collaborative filtering using a neural network**

*Erdman Alexander Alekseevich*

*Sholom-Aleichem Priamursky State University*

*Student*

### **Abstract**

The article discusses the development of a recommendation system based on collaborative filtering using a neural network. The Python and TensorFlow libraries in the Google Colaboratory environment are used for implementation. The result of the study is a model capable of predicting movie ratings for users and generating personalized recommendations.

**Keywords:** TensorFlow, Python, neural networks

## **1 Введение**

### **1.1 Актуальность**

Системы рекомендаций играют ключевую роль в современных информационных системах, помогая пользователям находить релевантные продукты, контент и услуги. Одним из популярных методов для создания таких систем является коллаборативная фильтрация, которая использует данные о предпочтениях пользователей для предсказания их будущих интересов. В данной статье рассматривается разработка системы рекомендаций на основе коллаборативной фильтрации с использованием нейронной сети. Для реализации используются библиотеки Python и TensorFlow в среде Google Colaboratory. Этот подход позволяет эффективно обрабатывать большие объемы данных и строить модели, способные

предсказывать предпочтения пользователей с высокой точностью. В статье также обсуждаются этапы предобработки данных, построения и обучения модели, а также оценка её производительности. В заключение представлены результаты экспериментов и возможные направления для дальнейшего улучшения системы.

### **1.2 Обзор исследований**

Д.И. Суханов в своей статье осветил вопрос реализации коллаборативного алгоритма рекомендации с обработкой и хранением разреженных матрицы [1]. Д.Е. Макарова в своей работе описала проектирование и разработку рекомендательной системы с коллаборативной фильтрацией для интернет-магазина [2]. Т.П. Рубцова и Ю.Д. Теплякова произвели обзор метода коллаборативной фильтрации, который является наиболее популярным и базовым методом построения предсказаний в рекомендательных системах [3]. Б.П. Гондран Альмега и Т.В. Батура предложили новый метод коллаборативной фильтрации под названием ARCADIUS [4]. Ч. Ма и М. Гао в совместном исследовании разработали стратегию рекомендаций на основе коллаборативной фильтрации для снижения ложных оценок, которая повышает точность рекомендаций по сравнению с использованием только одного алгоритма коллаборативной фильтрации, одновременно снижая влияние недостоверных оценок [5].

### **1.3 Цель исследования**

Целью данного исследования является разработка и практическое применение системы рекомендаций на основе коллаборативной фильтрации с использованием нейронной сети.

## **2 Материалы и методы**

Для разработки системы рекомендаций используется набор данных MovieLens, содержащий информацию о рейтингах фильмов, предоставленных пользователями. Методология включает предобработку данных, построение нейронной сети с использованием библиотеки TensorFlow. В качестве среды разработки выступает сервис Google Colaboratory.

## **3 Результаты и обсуждения**

Коллаборативная фильтрация является одним из наиболее распространенных и эффективных методов для создания систем рекомендаций. Основная идея этого метода заключается в использовании данных о предпочтениях пользователей для предсказания их будущих интересов. Коллаборативная фильтрация основана на предположении, что пользователи с похожими предпочтениями в прошлом будут иметь схожие предпочтения в будущем.

Существует два основных подхода к коллаборативной фильтрации: методы на основе соседства и методы на основе матричной факторизации. Методы на основе соседства включают user-based и item-based

коллаборативную фильтрацию. User-based коллаборативная фильтрация предполагает, что пользователи, которые имели схожие предпочтения в прошлом, будут иметь схожие предпочтения в будущем. Для предсказания рейтинга фильма для пользователя используются рейтинги, предоставленные другими пользователями, которые имеют схожие предпочтения. Item-based коллаборативная фильтрация предполагает, что фильмы, которые были оценены схожим образом разными пользователями, будут оценены схожим образом и в будущем. Для предсказания рейтинга фильма для пользователя используются рейтинги, предоставленные для других фильмов, которые были оценены схожим образом.

Методы на основе матричной факторизации предполагают, что рейтинги можно представить в виде произведения двух матриц: матрицы пользователей и матрицы фильмов. Каждая строка матрицы пользователей представляет собой вектор признаков пользователя, а каждая строка матрицы фильмов представляет собой вектор признаков фильма. Произведение этих матриц дает предсказанные рейтинги. Одним из наиболее известных алгоритмов матричной факторизации является Singular Value Decomposition (SVD), который разлагает матрицу рейтингов на произведение трех матриц: матрицы пользователей, диагональной матрицы сингулярных значений и матрицы фильмов.

В контексте данной статьи используется подход коллаборативной фильтрации на основе нейронной сети. Этот подход сочетает в себе элементы матричной факторизации и нейронных сетей для предсказания рейтингов фильмов для пользователей.

Коллаборативная фильтрация имеет несколько преимуществ и недостатков. Преимущества включают простоту реализации, высокую точность предсказаний и адаптивность к изменениям в предпочтениях пользователей. Однако, фильтрация может быть менее эффективной для новых пользователей или новых фильмов, для которых нет достаточного количества данных о рейтингах. Методы на основе соседства могут быть вычислительно затратными для больших наборов данных, так как требуют вычисления схожести между всеми пользователями или фильмами. В реальных наборах данных рейтинги могут быть разреженными, что может затруднить точное предсказание рейтингов.

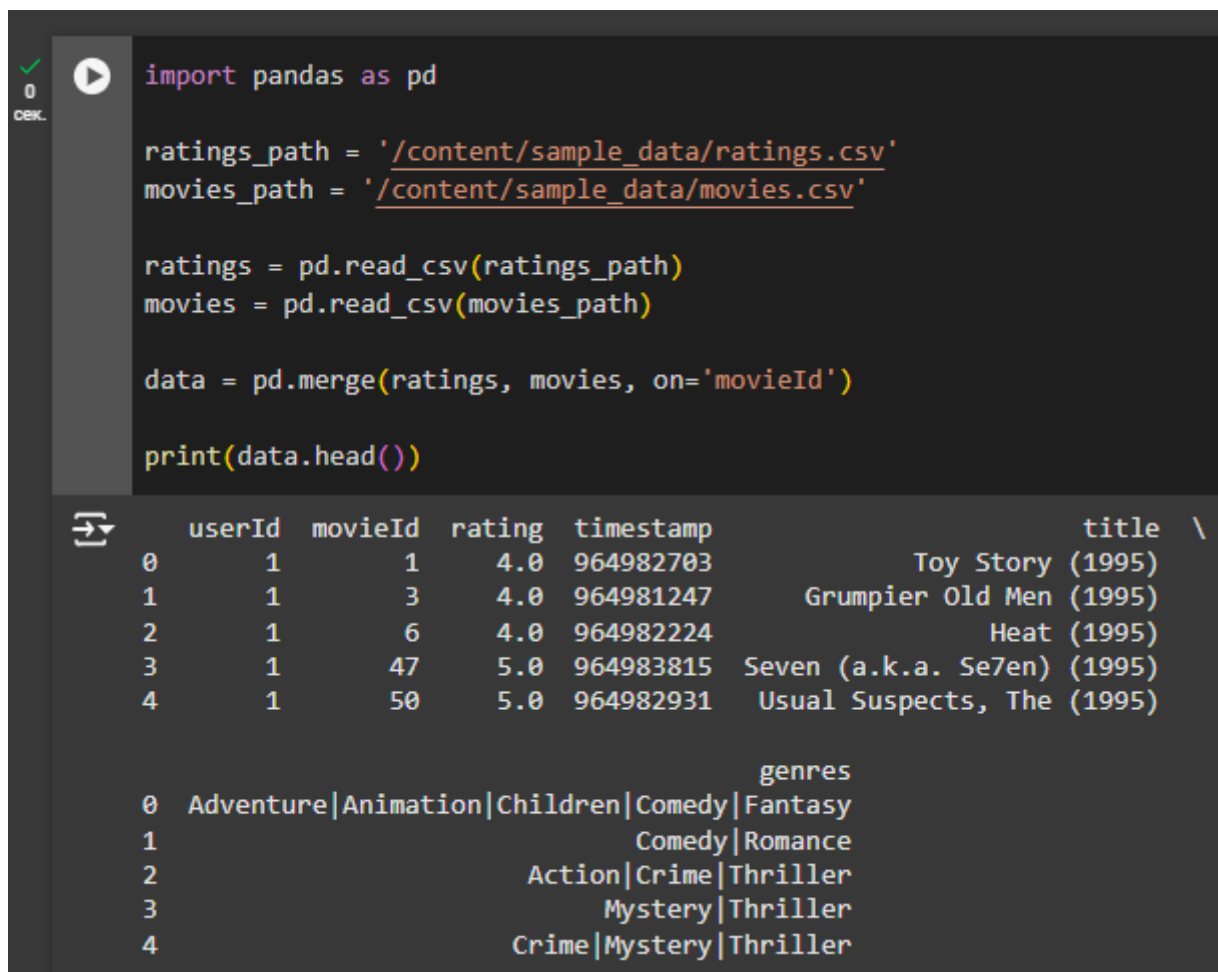
Для разработки системы рекомендаций используется набор данных “MovieLens”, который является одним из наиболее популярных и широко используемых наборов данных в области исследований систем рекомендаций. “MovieLens” содержит информацию о рейтингах фильмов, предоставленных пользователями, а также метаданные о фильмах, такие как названия и жанры. Данные предоставляются в формате CSV и включают в себя два основных файла: “ratings.csv” и “movies.csv”.

Файл “ratings.csv” содержит информацию о рейтингах, предоставленных пользователями. Каждая запись в этом файле включает в себя идентификатор пользователя (userId), идентификатор фильма (movieId), рейтинг (rating),

предоставленный пользователем, и временную метку (timestamp), указывающую время, когда рейтинг был предоставлен.

Файл “movies.csv” содержит метаданные о фильмах. Каждая запись в этом файле включает в себя идентификатор фильма (movieId), название фильма (title) и жанры фильма (genres).

Данные загружаются с помощью библиотеки “pandas”, которая обеспечивает удобные инструменты для работы с табличными данными (рис. 1).



```
import pandas as pd

ratings_path = '/content/sample_data/ratings.csv'
movies_path = '/content/sample_data/movies.csv'

ratings = pd.read_csv(ratings_path)
movies = pd.read_csv(movies_path)

data = pd.merge(ratings, movies, on='movieId')

print(data.head())
```

|   | userId | movieId | rating | timestamp | title \                     |
|---|--------|---------|--------|-----------|-----------------------------|
| 0 | 1      | 1       | 4.0    | 964982703 | Toy Story (1995)            |
| 1 | 1      | 3       | 4.0    | 964981247 | Grumpier Old Men (1995)     |
| 2 | 1      | 6       | 4.0    | 964982224 | Heat (1995)                 |
| 3 | 1      | 47      | 5.0    | 964983815 | Seven (a.k.a. Se7en) (1995) |
| 4 | 1      | 50      | 5.0    | 964982931 | Usual Suspects, The (1995)  |

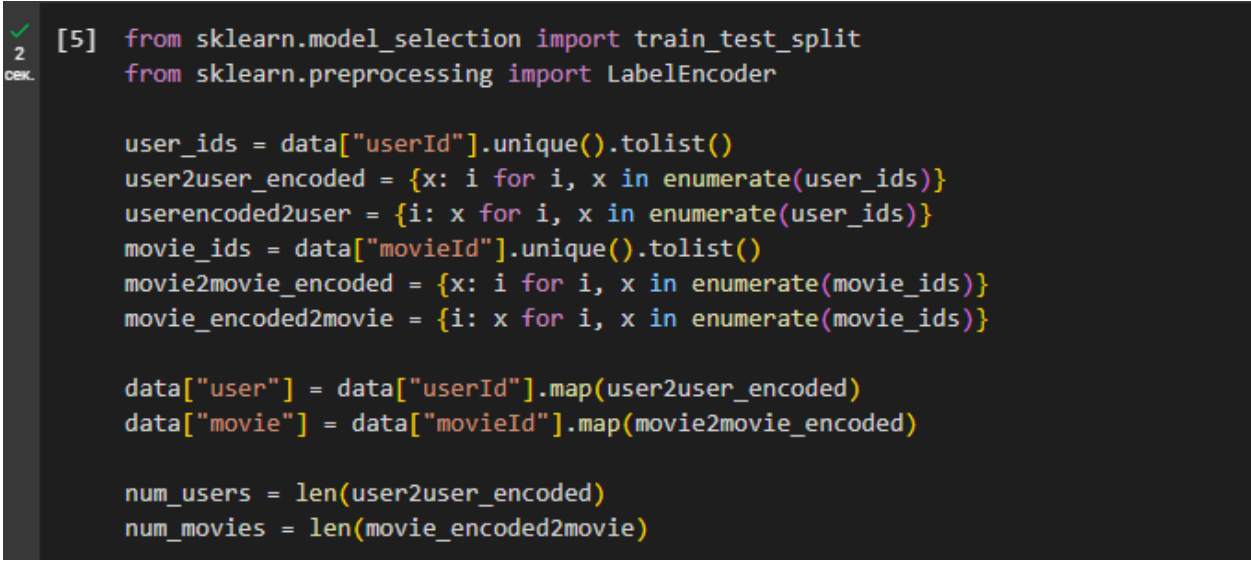
|   | genres                                      |
|---|---------------------------------------------|
| 0 | Adventure Animation Children Comedy Fantasy |
| 1 | Comedy Romance                              |
| 2 | Action Crime Thriller                       |
| 3 | Mystery Thriller                            |
| 4 | Crime Mystery Thriller                      |

Рисунок 1. Загрузка данных и объединение в единый DataFrame

Объединение данных в единый DataFrame является важным шагом в предобработке данных для создания системы рекомендаций. Этот процесс позволяет интегрировать различные источники данных, такие как информация о рейтингах пользователей и метаданные о фильмах, в единую структуру, что упрощает дальнейшую обработку и анализ данных. Причины данного объединения следующие: упрощение доступа к данным, унификация формата данных, обеспечение целостности данных и упрощение предобработки данных.

Следующим шагом является предобработка данных. Этот процесс включает в себя несколько ключевых шагов, которые обеспечивают качество и готовность данных для использования в модели машинного обучения. Для

использования данных в нейронной сети необходимо выполнить кодирование категориальных признаков, таких как идентификаторы пользователей и фильмов (рис. 2).

A screenshot of a code editor showing Python code for categorical encoding. The code imports `train_test_split` from `sklearn.model_selection` and `LabelEncoder` from `sklearn.preprocessing`. It then identifies unique user and movie IDs, creates mapping dictionaries, and applies them to the data. Finally, it counts the number of unique users and movies.

```
[5] from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import LabelEncoder

    user_ids = data["userId"].unique().tolist()
    user2user_encoded = {x: i for i, x in enumerate(user_ids)}
    userencoded2user = {i: x for i, x in enumerate(user_ids)}
    movie_ids = data["movieId"].unique().tolist()
    movie2movie_encoded = {x: i for i, x in enumerate(movie_ids)}
    movie_encoded2movie = {i: x for i, x in enumerate(movie_ids)}

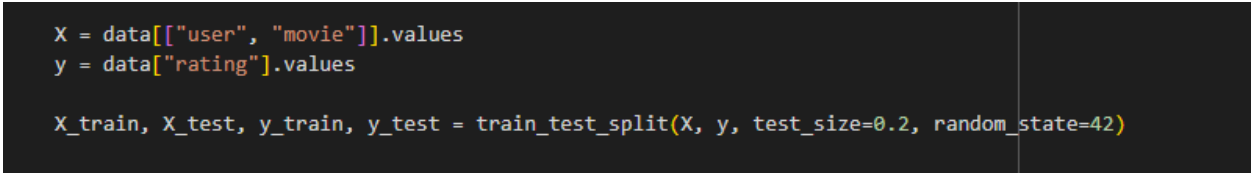
    data["user"] = data["userId"].map(user2user_encoded)
    data["movie"] = data["movieId"].map(movie2movie_encoded)

    num_users = len(user2user_encoded)
    num_movies = len(movie_encoded2movie)
```

Рисунок 2. Кодирование категориальных признаков

Это позволяет преобразовать категориальные данные в числовые значения, которые могут быть использованы в модели.

Для оценки производительности модели данные разделяются на обучающую и тестовую выборки (рис. 3).

A screenshot of a code editor showing Python code for splitting data. It extracts 'user' and 'movie' features into `X` and the 'rating' feature into `y`. Then, it uses `train_test_split` to divide the data into training and testing sets with a 0.2 test size and a fixed random state of 42.

```
X = data[["user", "movie"]].values
y = data["rating"].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Рисунок 3. Разделение данных на обучающую и тестовую выборки

Данное действие позволяет обучить модель на одной части данных и проверить её точность на другой части данных.

Построение модели является ключевым этапом в разработке системы рекомендаций. На этом этапе создается нейронная сеть, которая будет использоваться для предсказания рейтингов фильмов для пользователей. Архитектура нейронной сети включает в себя несколько ключевых компонентов: слои эмбедингов, слои Flatten, слои Concatenate и полносвязные слои. Эти компоненты позволяют моделировать сложные зависимости между пользователями и фильмами и предсказывать рейтинги с высокой точностью.

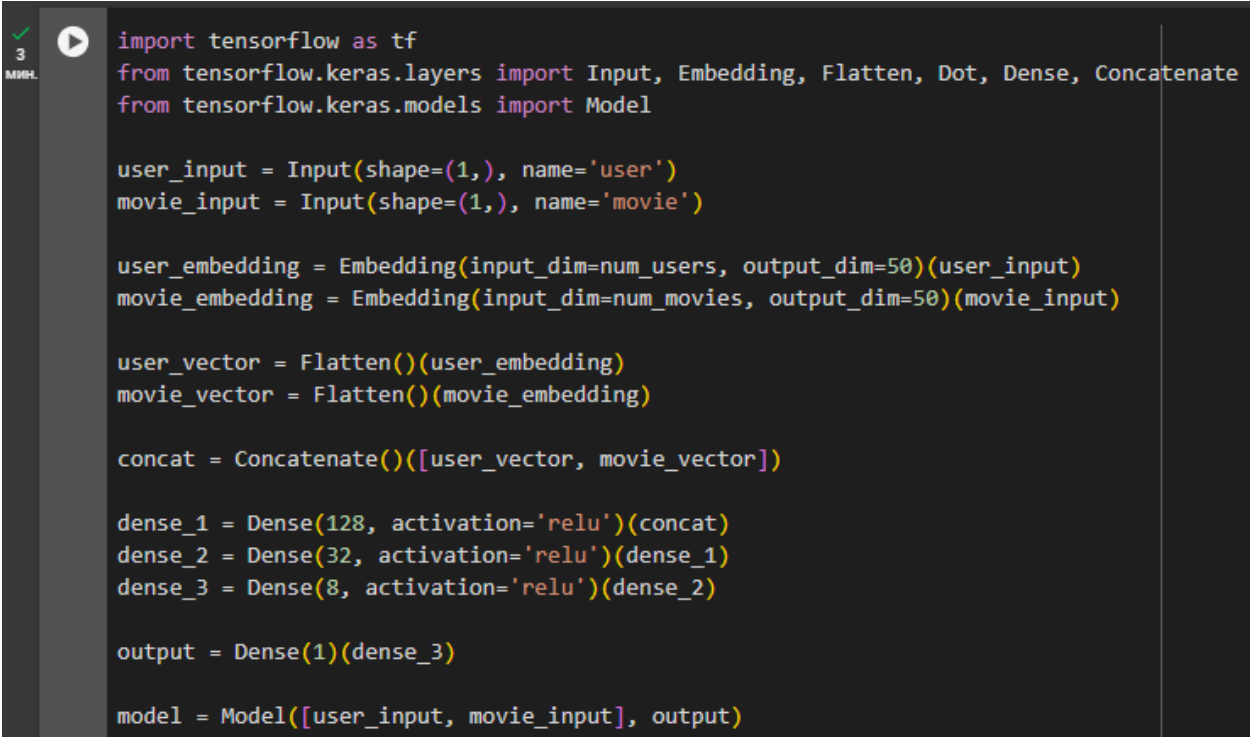
Слои эмбедингов используются для преобразования идентификаторов пользователей и фильмов в векторы признаков. Эти векторы признаков (эмбединги) позволяют моделировать сложные зависимости между пользователями и фильмами.

Слои Flatten преобразуют многомерные эмбединги в плоские векторы, которые могут быть использованы в полносвязных слоях.

Слои Concatenate объединяют векторы признаков пользователей и фильмов в единый вектор, который затем используется в полносвязных слоях.

Полносвязные слои используются для моделирования сложных зависимостей между пользователями и фильмами. Эти слои включают в себя несколько уровней нейронов, которые обрабатывают входные данные и предсказывают рейтинги фильмов для пользователей.

Для создания нейронной сети используется библиотека TensorFlow. Входные данные включают в себя идентификаторы пользователей и фильмов, которые преобразуются в векторы признаков с помощью слоев эмбедингов. Затем эти векторы признаков объединяются и проходят через несколько полносвязных слоев, которые предсказывают рейтинги фильмов для пользователей (рис. 4).



```
import tensorflow as tf
from tensorflow.keras.layers import Input, Embedding, Flatten, Dot, Dense, Concatenate
from tensorflow.keras.models import Model

user_input = Input(shape=(1,), name='user')
movie_input = Input(shape=(1,), name='movie')

user_embedding = Embedding(input_dim=num_users, output_dim=50)(user_input)
movie_embedding = Embedding(input_dim=num_movies, output_dim=50)(movie_input)

user_vector = Flatten()(user_embedding)
movie_vector = Flatten()(movie_embedding)

concat = Concatenate()([user_vector, movie_vector])

dense_1 = Dense(128, activation='relu')(concat)
dense_2 = Dense(32, activation='relu')(dense_1)
dense_3 = Dense(8, activation='relu')(dense_2)

output = Dense(1)(dense_3)

model = Model([user_input, movie_input], output)
```

Рисунок 4. Инициализация модели

После определения архитектуры нейронной сети модель компилируется с использованием оптимизатора Adam и функции потерь среднеквадратичной ошибки (MSE). Эти параметры обеспечивают эффективное обучение модели и минимизацию ошибки предсказаний.

Модель обучается на тренировочных данных в течение нескольких эпох (рис. 5). Это позволяет модели адаптироваться к данным и улучшить точность предсказаний. В процессе обучения модель также оценивается на валидационных данных для проверки её производительности.

```

history = model.fit(
    x=[X_train[:, 0], X_train[:, 1]],
    y=y_train,
    epochs=10,
    verbose=1,
    validation_data=([X_test[:, 0], X_test[:, 1]], y_test)
)

```

| Epoch       | 2521/2521 | Time         | acc    | loss   | val_acc | val_loss |
|-------------|-----------|--------------|--------|--------|---------|----------|
| Epoch 1/10  | 2521/2521 | 19s 7ms/step | 0.0265 | 1.8878 | 0.0276  | 0.7974   |
| Epoch 2/10  | 2521/2521 | 18s 7ms/step | 0.0279 | 0.6912 | 0.0276  | 0.7580   |
| Epoch 3/10  | 2521/2521 | 20s 7ms/step | 0.0278 | 0.6394 | 0.0276  | 0.7631   |
| Epoch 4/10  | 2521/2521 | 20s 7ms/step | 0.0272 | 0.5981 | 0.0276  | 0.7712   |
| Epoch 5/10  | 2521/2521 | 19s 7ms/step | 0.0287 | 0.5558 | 0.0276  | 0.7861   |
| Epoch 6/10  | 2521/2521 | 22s 7ms/step | 0.0278 | 0.4972 | 0.0276  | 0.8017   |
| Epoch 7/10  | 2521/2521 | 19s 7ms/step | 0.0278 | 0.4378 | 0.0276  | 0.8394   |
| Epoch 8/10  | 2521/2521 | 20s 7ms/step | 0.0280 | 0.3943 | 0.0276  | 0.8422   |
| Epoch 9/10  | 2521/2521 | 17s 7ms/step | 0.0276 | 0.3513 | 0.0275  | 0.8647   |
| Epoch 10/10 | 2521/2521 | 20s 7ms/step | 0.0272 | 0.3196 | 0.0273  | 0.8918   |

Рисунок 5. Обучение модели

В процессе обучения модели, значения “loss” и “val\_loss” (значения функций потерь) обычно выводятся для каждой эпохи. Это позволяет отслеживать, как модель улучшает свои предсказания на тренировочных данных и как она обобщает эти знания на валидационных данных. Если значение “loss” уменьшается с каждой эпохой, это означает, что модель улучшает свои предсказания на тренировочных данных. Если значение “val\_loss” также уменьшается, это означает, что модель хорошо обобщает данные и предсказывает значения на новых данных. Если “val\_loss” начинает увеличиваться, это может указывать на переобучение (overfitting), когда модель слишком хорошо подстраивается под тренировочные данные и теряет способность обобщать. Точность измеряет долю правильно предсказанных меток классов. Высокое значение точности указывает на хорошую производительность модели.

Для оценки производительности модели используется тестовая выборка. Модель оценивается по метрике среднеквадратичной ошибки (MSE) (рис. 6).

```

loss, accuracy = model.evaluate(x=[X_test[:, 0], X_test[:, 1]], y=y_test)
print(f"Loss: {loss}")
print(f"Accuracy: {accuracy}")

```

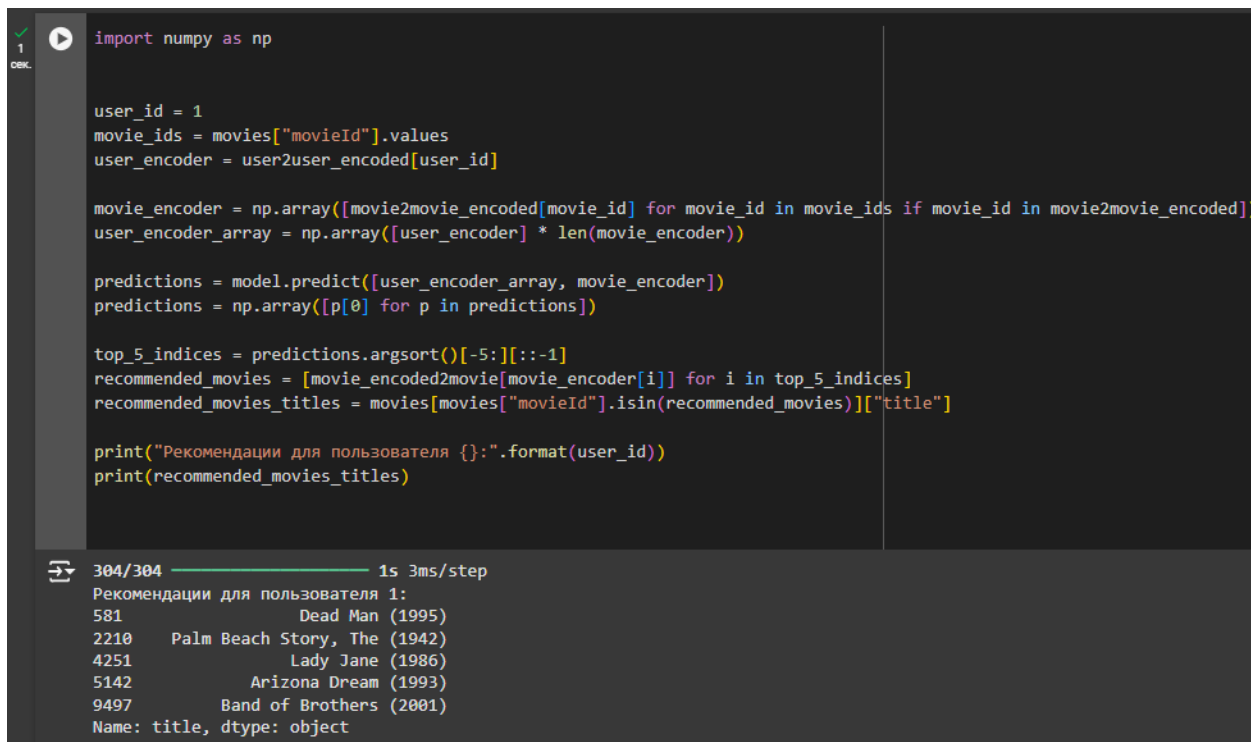
| Progress | Time        | acc    | loss   |
|----------|-------------|--------|--------|
| 631/631  | 1s 2ms/step | 0.0281 | 0.8986 |

Loss: 0.8917862176895142  
Accuracy: 0.027320507913827896

Рисунок 6. Оценка модели

Следующим шагом является генерация рекомендаций для пользователей (рис. 7). Генерация рекомендаций включает в себя несколько

ключевых шагов. Во-первых, используя обученную модель, предсказываются рейтинги для всех фильмов для конкретного пользователя. Это позволяет определить, какие фильмы пользователь, вероятно, оценит высоко. Во-вторых, после предсказания рейтингов, фильмы сортируются по предсказанным рейтингам в порядке убывания. Из списка выбираются топ-N фильмов, которые и будут рекомендованы пользователю.



```
import numpy as np

user_id = 1
movie_ids = movies["movieId"].values
user_encoder = user2user_encoded[user_id]

movie_encoder = np.array([movie2movie_encoded[movie_id] for movie_id in movie_ids if movie_id in movie2movie_encoded])
user_encoder_array = np.array([user_encoder] * len(movie_encoder))

predictions = model.predict([user_encoder_array, movie_encoder])
predictions = np.array([p[0] for p in predictions])

top_5_indices = predictions.argsort()[-5:][::-1]
recommended_movies = [movie_encoded2movie[movie_encoder[i]] for i in top_5_indices]
recommended_movies_titles = movies[movies["movieId"].isin(recommended_movies)]["title"]

print("Рекомендации для пользователя {}:".format(user_id))
print(recommended_movies_titles)
```

304/304 ————— 1s 3ms/step  
 Рекомендации для пользователя 1:  
 581 Dead Man (1995)  
 2210 Palm Beach Story, The (1942)  
 4251 Lady Jane (1986)  
 5142 Arizona Dream (1993)  
 9497 Band of Brothers (2001)  
 Name: title, dtype: object

Рисунок 7. Генерация рекомендаций

Сначала выбирается идентификатор пользователя, для которого будут генерироваться рекомендации. Затем осуществляется проверка, что все идентификаторы фильмов присутствуют в словаре “movie2movie\_encoded”. Это необходимо для предотвращения ошибок при предсказании рейтингов. Далее создается массив, содержащий идентификатор пользователя для каждого фильма. Это позволяет модели предсказывать рейтинги для всех фильмов для конкретного пользователя. Используя обученную модель, предсказываются рейтинги для всех фильмов для конкретного пользователя. Предсказанные рейтинги сортируются в порядке убывания, и выбираются топ-5 фильмов с наивысшими рейтингами. Эти фильмы и будут рекомендованы пользователю.

Данный результат был получен при значении функции потерь среднеквадратичной ошибки (MSE) равным 0.89 (идеальное значение составляет 0.0) и использования оптимизатора Adam. Результат функции потерь говорит о том, что показатель точности модели можно улучшить.

Чтобы добиться более производительной модели создаются дополнительные слои “Dropout” после каждого слоя “Dense” для предотвращения переобучения (рис. 8). Данный слой случайным образом

отключает часть нейронов во время обучения, что помогает модели обобщать лучше.

```
dense_1 = Dense(128, activation='relu')(concat)
dense_1 = Dropout(0.5)(dense_1)
dense_2 = Dense(64, activation='relu')(dense_1)
dense_2 = Dropout(0.5)(dense_2)
dense_3 = Dense(32, activation='relu')(dense_2)
dense_3 = Dropout(0.5)(dense_3)
```

Рисунок 8. Добавление слоев “Dropout”

Коэффициент выброса нейронов равен 0.5, это означает, что при каждом проходе через слой “Dropout” 50% нейронов будут случайным образом отключены. Это помогает предотвратить переобучение, так как модель не может полагаться на конкретные нейроны и вынуждена учиться более обобщенным представлениям данных.

Следующим шагом улучшения производительности модели является увеличение количества эпох с 10 до 30 для более длительного обучения модели. Это может помочь модели лучше сходиться к оптимальному решению.

Для того, чтобы модели лучше захватывать сложные зависимости в данных, Изменяется количества нейронов в слоях “Dense” для более сложной архитектуры модели (рис. 8).

Также для улучшения результата модели был расширен список данных в каждом файле.

Чтобы сделать модель более устойчивой к выбросам в данных, используется средняя абсолютная ошибка вместо средней квадратичной. Использование метрики “mean\_absolute\_error” вместо “accuracy” в задаче связано с тем, что “accuracy” обычно используется для задач классификации, где целью является предсказание дискретных классов. В задаче регрессии, такой как предсказание рейтингов фильмов, целью является предсказание непрерывных значений, и “accuracy” является менее эффективной метрикой.

Наравне со сменой метрики меняется и оптимизатор для обучения нейронной сети на “RMSprop”. Данный метод может быть предпочтительнее из-за своей стабильности и простоты, что помогает предотвратить колебания в процессе оптимизации и упрощает настройку гиперпараметров. “RMSprop” адаптирует скорость обучения для каждого параметра на основе среднего квадрата градиентов. Причина замены “Adam”, который является мощным оптимизатором, но иногда может быть слишком агрессивным и приводить к переобучению или нестабильному обучению. В контексте рассматриваемой задачи “RMSprop” должен показать лучшие результаты.

После внесения изменений в модели происходит повторное обучение и испытание (рис. 9).

```
Mean Absolute Error: 0.6440149545669556
780/780 ————— 7s 8ms/step
Рекомендации для пользователя 1:
108                                Braveheart (1995)
351                                Forrest Gump (1994)
522                                Schindler's List (1993)
7029    Lord of the Rings: The Return of the King, The...
41017                                Planet Earth (2006)
Name: title, dtype: object
```

Рисунок 9. Результат модели после внесённых изменений

На результате видно, что средняя абсолютная ошибка равна 0.64, что в свою очередь говорит о том, что это хороший результат, который указывает на то, что модель предсказывает рейтинги с достаточной точностью.

#### 4 Вывод

Таким образом в результате исследования была создана система рекомендаций на основе коллаборативной фильтрации с использованием нейронной сети. После первого тестирования модели были внесены изменения в метриках и оптимизаторе, что сделало модель более производительной и привело к более точным результатам выбора фильма для пользователя.

#### Библиографический список

1. Суханов Д.И. Реализация коллаборативной фильтрации в сочетании с разреженными матрицами // В сборнике: Управление большими системами. сборник научных трудов XIX Всероссийской школы-конференции молодых ученых. Воронеж, 2023. С. 244-247.
2. Макарова Д.Е. Разработка рекомендательной системы с коллаборативной фильтрацией // В сборнике: Научное сообщество студентов XXI столетия. Технические науки. сборник статей по материалам СХХХII студенческой международной научно-практической конференции. Новосибирск, 2023. С. 47-56.
3. Рубцова Т.П., Теплякова Ю.Д. Обзор метода коллаборативной фильтрации разработки рекомендательных систем // В сборнике: Фундаментальные и прикладные научные исследования: актуальные вопросы, достижения и инновации. сборник статей LXIX Международной научно-практической конференции. Пенза, 2024. С. 11-13.
4. Гондран Альмега Б.П., Батура Т.В. Реализация алгоритма на основе коллаборативной фильтрации для item-based-подхода // Сборник избранных статей научной сессии ТУСУР. 2023. № 1-2. С. 184-186.
5. Ма Ч., Гао М. Стратегия рекомендаций на основе коллаборативной фильтрации для снижения ложных оценок // В сборнике: Актуальные проблемы авиации и космонавтики. Сборник материалов IX Международной научно-практической конференции, посвященной Дню космонавтики. В 3-х томах. Красноярск, 2023. С. 74-77.