Wir haben ein Empfehlungssystem entwickelt, das den Content-based Algorithm mit dem User Collaborative Algorithm kombiniert. Unser Ziel ist es, dem eingeloggten Nutzer Filme vorzuschlagen, die auch den Präferenzen anderer ähnlicher Nutzer entsprechen, wobei wir insbesondere auf Übereinstimmungen in den Genre-Kategorien achten.

Das Ganze läuft wie folgt ab:

1.Der Content-based Algorithm:

Um die individuellen Vorlieben der Nutzer zu berücksichtigen und basierend darauf Filmempfehlungen anzuzeigen, verwenden wir einen Inhaltsbasierten Algorithmus.

Dies beginnt damit, dass die von dem Nutzer favorisierten Filme in einer Liste, der Favoritenliste gespeichert werden.

Danach muss mithilfe des Filmtitels herausgefunden werden, welchen Genres die Favoriten angehören.

Hierfür wird in der JSON-Datei mit den Filmdaten nach dem Titel des favorisierten Filmes gesucht und dann das dazugehörige Genre ausgelesen. Es ist zu beachten, dass ein Film auch mehrere Genres haben kann. In diesem Fall wird jedes einzelne Genre ausgelesen.

Dieses Vorgehen wird auf jeden Film aus der Favoritenliste angewendet.

Die zugeordneten Genres werden nun in eine neue List, die Genreliste geschrieben. Im darauffolgenden Schritt sollen nun ähnliche Filme herausgefiltert werden, welche dem Nutzer gefallen könnten.

Dies kann durch einen Genre Counter realisiert werden. Dabei wird gezählt, wie häufig einzelne Genres in der Favoritenliste vertreten sind. Wenn beispielsweise zwei Filme dem Genre „Drama“ zugeordnet sind, wird die Anzahl für „Drama“ entsprechend um zwei erhöht.

Falls ein Film mehreren Kategorien angehört, beispielsweise „Action“ und „Drama“, wird für beide Genres der Zähler jeweils um eine Stelle hochgezählt.

Anschließend wird geprüft, ob ein Film bereits in der Favoritenliste enthalten ist, wenn das der Fall ist, werden solche Filme nicht erneut als Empfehlung angezeigt.

Um zu ermöglichen die Empfehlungen nach Relevanz zu ordnen, arbeiten wir mit einem Score. Dieser gibt an, wie viele Übereinstimmungen zwischen den bevorzugten Genres des Nutzers und den Filmen aus der allgemeinen JSON-Filmdatenbank existieren. Der Film mit der höchsten Übereinstimmung wird an erster Stelle in der Empfehlungsliste angezeigt. Die resultierenden Empfehlungen werden schließlich in einer Liste gespeichert und dem Nutzer ausgegeben. Das bedeutet im Klartext: Je höher der Score, desto weiter oben erscheint der Film in der Liste.

Durch dieses Verfahren bietet das Empfehlungssystem präzise und personalisierte Vorschläge, die auf den individuellen Filmgeschmack des Nutzers abgestimmt sind.

Der Algorithmus grafisch dargestellt:

Favoritenliste

Score

Übereinstimmung

Analyse

Genre von Favoriten aus JSON

Counter:

Drama : 2

Action : 2

Romantik : 1

Genreliste:

Drama

Action

Romantik

Drama

JSON

Datei

Genre Counter

Empfehlungsliste

Für carina als Bild:

Ein Bild, das Text, Screenshot, Diagramm, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

2.Der User Collaborative Algorithm:

Damit den Nutzern zusätzlich Filme empfohlen werden können, welche auch deren Freunden gefallen, verwenden wir einen benutzerbasierenden Algorithmus.

Im Code beginnen wir damit, die Daten der Filme und der gesamten Nutzerbasis zu laden und in entsprechende Variablen zu speichern. Anschließend ermitteln wir für jeden Nutzer dessen bevorzugte Genres und speichern diese als separate Listen. Für den eingeloggten Nutzer extrahieren wir dessen Genres und favorisierten Filme.

Danach durchlaufen wir alle anderen Nutzer, wobei der eingeloggte Nutzer dabei ausgeschlossen wird. Für jeden dieser Nutzer überprüfen wir die Ähnlichkeit zum eingeloggten Nutzer und fügen diesen Wert einer speziellen Ähnlichkeitsliste hinzu.

Mithilfe der Jaccard-Formel können wir die Ähnlichkeitswerte für die Benutzer erstellen und danach ansteigend nach dem Wert sortieren, sodass der ähnlichste Nutzer an erster Stelle steht.

Die Jaccard-Formel oder auch jaccard simularity dient zur Berechnung von Ähnlichkeiten zweier Mengen. Sie wird definiert als das Verhältnis der Größe der Schnittmenge zur Größe der Vereinigungsmenge dieser beiden Mengen:

J(A, B) **=** | A∩B | / | A∪B |

| A∩B | ist die Anzahl der gemeinsamen Elemente (Schnittmenge).

∣A∪B∣ ist die Anzahl der insgesamt vorhandenen unterschiedlichen Elemente (Vereinigungsmenge).

In unserem System wäre das zum Beispiel wenn zwei Nutzer folgenden Lieblingsfilmen haben:

Nutzer A: { Romantik, Action, Drama } und

Nutzer B: { Action, Drama, Thriller }

Hier wäre die Schnittmenge {Action, Drama } und bekommt dadurch die Größe 2 und die Vereinigungsmenge wäre {Romantik, Action, Drama, Thriller}, somit die Größe 4. Wenn man das in die Formel einsetzt rechnet man 2/4 und bekommt einen Wer von 50% heraus. Dies sagt uns, dass die Jaccard-Ähnlichkeit 0,5 (also 50 %) beträgt.

Sie kann zwischen 0 und 1 liegen und je höher die Zahl, desto ähnlicher sind die beiden Datensätze.

Im Anschluss entnehmen wir diesem ähnlichsten Nutzer dann dessen Filmempfehlungen, die anschließend an den Controller zurückgegeben werden.

Der Algorithmus grafisch dargestellt:

Andere Nutzer

Aktueller Nutzer

Filme aus Favoritenliste

Filme aus Favoritenliste

Genreliste

Genreliste

Ähnlichkeitsbestimmung

(jaccard simularity)

Ähnlichkeitsliste sortiert

Als Bild:

Ein Bild, das Text, Screenshot, Diagramm, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

Quelle für jaccard formel: <https://statologie.de/jaccard-koeffizient-python/>

Zukunft

Für die zukünftige Entwicklung unseres Projektes konzentrieren wir uns auf die Optimierung und Erweiterung des Systems. Zunächst würden wir bestehende Bugs beheben, um das Programm möglichst zuverlässig und fehlerfrei zu machen.

Des Weiteren würden wir anstatt der JSON-Dateien eine richtige Datenbank (z.B. SQL) implementieren. Dies würde eine automatisierte Aktualisierung der Filmdaten ermöglichen, um den Nutzern stets die neusten Informationen zu garantieren. Außerdem würde es Anpassungen des Programms vereinfachen, da wir Informationsneuheiten nicht mehr manuell eintragen müssten. Ein zusätzlicher Vorteil der Verwendung einer Datenbank liegt im Speicherplatz, da JSON-Dateien aufgrund ihrer alphanummerischen Darstellung mehr Speicherplatz beanspruchen, als binäre Datenformate. Ein weiterer Nebeneffekt wäre, dass wir dadurch die Performance unseres Programms verbessern. Durch die Umstellung kann der Speicherbedarf reduziert und die Performance des Programms wesentlich verbessert werden.

Darüber hinaus ermöglicht eine Datenbank mehreren Benutzern gleichzeitig auf die Informationen zuzugreifen, weshalb es das System vielseitiger nutzbar und noch benutzerfreundlicher macht. Zusätzlich ist geplant zu den jeweiligen Filmen auch die Cover ausgeben zu können

Eine weitere geplante Verbesserung wäre die Integration sozialer Funktionen. Nutzer sollen die Möglichkeit haben Freunde zu „adden“, wodurch auch Empfehlungen basierend auf den Vorlieben von Freunden erscheinen. Dadurch entstehen noch individuellere Empfehlungen.

Um die Reichweite und Zugänglichkeit haben wir uns überlegt unser Empfehlungssystem so zu erweitern, damit wir es plattformübergreifend zur Verfügung stellen können. Das bedeutet, dass dem Benutzer nicht nur Filmempfehlungen von einem bestimmten Streamingdienst angezeigt werden, sondern von vielen verschiedenen. Dabei soll natürlich auch deutlich sein, auf welcher Plattform der Film angeschaut werden kann.

Für zukünftige, größere Nutzerdatenbanken planen wir den Einsatz von Machine Learning (ML). Diese Technologie könnten uns dabei helfen, Informationen effizienter und schneller zu verarbeiten. Zum jetzigen Zeitpunkt wäre der Aufwand für die Implementierung von ML – einschließlich zusätzlicher Treiber, Software und Bibliotheken – jedoch unverhältnismäßig hoch und würde sich angesichts unserer aktuellen Datenbankgröße nicht lohnen.

Ähnliches gilt für die Integration einer dedizierten Datenbank wie SQL. Während SQL-Datenbanken für große Datenmengen ideal sind, hätte ihre Einbindung in unserem derzeitigen Stadium die Entwicklung erschwert. Für Entwickler kann es aufwendiger sein, SQL-Datenbankdateien zu lesen und Fehler direkt in der Datenbank zu beheben. Dies hätte wiederum Konsequenzen für die Entwicklung und Fehlerbehebung des Algorithmus haben können. Eine SQL-Datenbank wird erst dann sinnvoll, wenn unsere Datenmengen signifikant anwachsen.