Recommender Systems

Marius Hirt
Fakultät Informatik
Hochschule Furtwangen
Furtwangen, Deutschland
marius.hirt@hs-furtwangen.de

Jan Knoblauch
Fakultät Informatik
Hochschule Furtwangen
Furtwangen, Deutschland
jan.alexander.knoblauch@hs-furtwangen.de

***Zusammenfassung*—Empfehlungssysteme sind aus dem modernen Web nicht wegzudenken. Für Unternehmen können mit relevanten Empfehlungen enorme Summen an Geschäftswert generiert werden. Für Benutzer wird dabei ein personalisiertes Erlebnis auf der Plattform geboten. In dieser Arbeit werden zunächst die Herausforderungen an Empfehlungssysteme sowie die unterschiedlichen Arten von Algorithmen umrissen. In einem Proof-of-Concept (PoC) werden zwei verfügbare Lösungen – Amazon Personalize und LibRec – untersucht und bewertet.**

Grundlage des PoC ist ein offline Wikipedia-Modell, welches eine ausgewählte Untermenge von Artikeln aus unterschiedlichen Kategorien in einem zusammenhängenden Graphen abbildet. Zur Generierung von Daten für das Empfehlungssystem wird der Graph von simulierten Benutzern zufällig durchlaufen. Jeder Nutzer bekommt dabei pro Sitzung eine Lieblingskategorie. Artikel aus dieser Kategorie werden sehr gut bewertet, im Graph weit entfernte Artikel bekommen in dieser Sitzung schlechtere Bewertungen. Das Empfehlungssystem soll die Zugehörigkeit der Artikel zu den Kategorien reproduzieren, ohne explizite Kenntnis der Kategorien zu haben.

Als Bewertungskriterium werden für jeden Artikel die besten fünf Empfehlungen des jeweiligen Empfehlungssystems betrachtet. Als Metrik dient dabei der Prozentsatz, wie viele der Empfehlungen zur selben Kategorie gehören, wie der Artikel, für den die Empfehlungen gegeben werden. Amazon Personalize erreicht mit diesem Test eine Präzision von über 95%, LibRec erreicht dabei bis zu 73%.

I. Einleitung

Empfehlungssysteme stecken nicht nur hinter offensichtlichen Funktionen wie dem Vorschlag, welche Artikel in einem Online-Shop für den Kunden außerdem interessant sein könnten, sondern sind in sehr vielen alltäglichen Anwendungen und Diensten integriert. Empfehlungssysteme der Marktführer Google, YouTube, Spotify, Facebook, Instagram, Amazon und weiteren Anbietern erstellen meistens sehr genaue Profile ihrer Nutzer. Dadurch steuern sie was wir bei einer Google-Suche finden, welche Musik und Podcasts wir hören, was wir auf sozialen Plattformen sehen und welche Artikel wir online kaufen. Das alles wirkt sich stark auf uns aus und bindet uns langfristig an Plattformen, die uns gut kennen. Zwar werden in Europa durch die Datenschutz-Grundverordnung Unternehmen in der Weitergabe von persönlichen Daten eingeschränkt. Erlaubt ist es ihnen durch richtige Datenschutzerklärungen jedoch weiterhin. In der Branche der Online-Dienste machen präzise Nutzerdaten und gute Empfehlungssysteme heutzutage oftmals einen Großteil des Wertes eines Unternehmens aus.

Schätzungen zufolge beläuft sich der Traffic, der durch das Empfehlungssystem von Amazon generiert wird, auf bis zu 25% der gesamten Seitenaufrufe der Webseite [1]. Der Streaming-Anbieter Netflix behauptet, dass 80% aller Filme, die bei dem Anbieter geschaut werden, von ihrem System vorgeschlagen wurden [2]. Der geschätzte Geschäftswert von Netflix' Empfehlungssystem liege damit bei über 1 Mrd. US-Dollar.

Empfehlungssysteme halten Nutzer bei Streaming-Anbietern und sozialen Netzwerken auf der Plattform, die ohne interessanten Inhalte die Plattform – kostenlos oder kostenpflichtig – verlassen würden. Bei Online-Shops dienen solche Systeme zudem der Präsentation des Sortimentes. Niemand kann die Produktauswahl bei Händlern wie Amazon überblicken, und kaum jemand vergleicht alle angebotenen Artikel über mehrere hundert Seiten. Viele Online-Shops sind mit Systemen ausgestattet, die interessante Artikel zuerst anzeigen, damit potentielle Kunden ihre gewünschten Artikel schneller finden. Die Reihenfolge der Artikel entwickelt sich dabei dynamisch anhand neuster Trends. Außerdem werden oftmals ergänzende Artikel identifiziert, wie beispielsweise passende Batterien zu einer Taschenlampe („Cross-Selling"). Aber auch teurere Artikel bzw. Artikel mit einer höheren Marge aus Sicht des Händlers werden vorgeschlagen, um den Profit zu steigern („Upselling").

Empfehlungssysteme sind also keine optionale Verbesserung für vorhandene Dienste, sondern in ganz vielen Fällen ein essentieller Bestandteil. Doch wie funktionieren Empfehlungssysteme? Können nur große IT-Unternehmen von solchen Systemen profitieren? Welche Möglichkeiten und Grenzen bieten verfügbare Systeme?

Diese Arbeit gibt zuerst einen grundlegenden Einblick in den Aufbau von Empfehlungssystemen. Anschließend wird das Konzept sowie zwei Empfehlungssysteme, die anhand des Konzeptes verglichen wurden, vorgestellt. Anschließend wird auf die Umsetzung des Konzeptes sowie auf die erzielten Ergebnisse eingegangen. Daraufhin werden weitere Systeme genannt, die im Rahmen dieser Arbeit aber nicht weiter untersucht werden konnten. Abschließend werden die gewonnenen Eindrücke in einem Fazit zusammengefasst.

A. Herausforderungen für Empfehlungssysteme

Eines der am längsten bestehenden Empfehlungssysteme ist beim Onlineversand Amazon zu finden. Seit der ersten Veröffentlichung ihres Algorithmus [3] hat sich aus dem damaligen Bücherhandel der heutige Versandhandel mit der wahrscheinlich vielfältigsten Produktpalette der Welt entwickelt. Durch diese jahrelange Entwicklung wurden wertvolle Erfahrungen gesammelt, die in [4] zusammengefasst werden. Einige der dort genannten Herausforderungen von Empfehlungssystemen werden hier kurz wiedergegeben.

Die Aufgabe eines Empfehlungssystems ist das Identifizieren von Beziehungen zwischen Artikeln eines Katalogs. Diese Beziehungen werden aus dem Benutzerverhalten abgeleitet, also wie ein Benutzer mit den unterschiedlichen Artikeln interagiert. Dabei ist immer

wichtig, den Kontext zu betrachten, in dem die Interaktion stattfindet.

Ein wichtiger Aspekt ist dabei die zeitliche Nähe zwischen einzelnen Interaktionen. Wenn zwei Bücher etwa innerhalb eines kurzen Zeitraumes gekauft werden, ist das ein gutes Indiz dafür, dass sie füreinander relevant sind. Liegen zwischen dem Kauf zweier Bücher aber mehrere Monate, kann über die Beziehung keine Aussage mehr getroffen werden.

Auch die zeitliche Abfolge spielt eine wichtige Rolle. Wenn ein Nutzer beispielsweise eine Digitalkamera kaufen möchte, ist es normal, dass dieser zunächst verschiedene Digitalkameras anschaut und sie miteinander vergleicht. Es ist zu diesem Zeitpunkt daher sinnvoll, Empfehlungen für weitere Digitalkameras zu geben. Nachdem ein Kauf abgeschlossen wurde ist es dagegen unwahrscheinlich, dass in absehbarer Zeit eine weitere Digitalkamera gekauft wird. Stattdessen sollten dann komplementäre Empfehlungen gegeben werden, wie etwa eine Speicherkarte. Umgekehrt ist es weniger sinnvoll nach dem Kauf einer Speicherkarte eine Digitalkamera zu empfehlen. Diese Beziehung geht also nur in eine Richtung.

Auf lange Sicht ist es auch wünschenswert die Interessen der Kunden bei Empfehlungen zu berücksichtigen, statt nur deren unmittelbare Intention. Durch den Kauf von Hobbyartikeln lässt sich so relativ leicht auf die Hobbys des Kunden schließen. Beim Kauf einer Digitalkamera kann man davon ausgehen, dass der Kunde ein langfristiges Interesse für Fotografie hat. Falls die unmittelbare Intention eines Kunden noch nicht fest steht, dann können Empfehlungen zu den Interessen des Kunden gegeben werden. Für generische Artikel, wie etwa ein Paar Socken, lassen sich solche Schlüsse nicht ziehen. Solche Produkte sind in jedem Haushalt zu finden und sollten für langfristige Empfehlungen nicht berücksichtigt werden.

Diese langlebigen Interessen können sich im Laufe der Zeit aber auch verändern. Im Optimalfall kann man auf diese Veränderungen nicht nur reagieren, sondern sie sogar vorhersehen. Beispielsweise lässt der Kauf von Babyartikeln darauf schließen, dass der entsprechende Kunde ein Elternteil eines Babys ist. Darauf aufbauend kann man dem Kunden über Jahre hinweg Empfehlungen geben, was das Kind im entsprechenden Alter brauchen könnte, wie beispielsweise ein Kinderfahrrad.

Diese Konzepte sind nicht nur auf physische Artikel aus Online-Shops anwendbar. Für Videos auf YouTube oder Artikel auf Nachrichtenseiten haben Benutzer ebenfalls kurz- und langfristige Interessen. Die Argumentation gilt also für Produktkataloge verschiedener Arten.

B. Methodiken von Empfehlungssystemen

Empfehlungssysteme arbeiten nach verschiedenen Vorgehensweisen. Unter inhaltsbasierten Empfehlungen („Content-based Filtering“) versteht man das Prinzip, bei dem vorhandene und zu empfehlende Inhalte wie beispielsweise Produkte in einem Online-Shop oder Filme eines Streaming-Anbieters gruppiert werden. Interessiert sich ein Nutzer für einen Inhalt einer bestimmten Gruppe, werden ihm andere Inhalte aus derselben Gruppe vorgeschlagen. Kennt ein Empfehlungssystem mit der Zeit mehrere solcher Gruppen, für die sich ein Nutzer interessiert, werden ihm Inhalte aus den jeweiligen Gruppen empfohlen, oder Inhalte, die mehreren dieser Gruppen zugeordnet sind [5].

Wohingegen bei diesem Ansatz die Inhalte gruppiert werden, verfolgt das Prinzip der kollaborativen Empfehlungen („Collaborative Filtering“) auf der Gruppierung von Nutzern. Ziel ist es, Nutzer mit ähnlichen Interessen zu identifizieren. Die Inhalte werden dabei nicht eingeteilt oder in Kategorien zusammengefasst. Sollen Empfehlungen für einen Nutzer generiert werden, wird ausgewertet, für welche Inhalte sich andere Nutzer aus dessen Gruppe interessiert haben.

Die Einteilungen von Inhalten und Nutzern sind in einem Empfehlungssystem in aller Regel nicht real nachvollziehbar sondern sind abstrakte Ergebnisse eines Modells, welches durch maschinelles Lernen erzeugt, erweitert und verbessert wird. Beide Konzepte haben ein Problem, wenn neue Inhalte oder neue Nutzer hinzukommen. Es ist nachvollziehbar, dass Empfehlungen schwierig sind, wenn über einen Inhalt oder einen Nutzer kaum Informationen vorliegen. Daher ist das ständige Lernen solcher Systeme eine wichtige Eigenschaft.

Beide vorgestellten Prinzipien sowie noch weitere Methodiken können sich gegenseitig gut ergänzen, sodass oftmals eine Mischung eingesetzt wird. Dabei wird unterschieden, ob die Prinzipien getrennt voneinander angewendet und die Ergebnisse daraus kombiniert werden, oder ob die Prinzipien in einem einzelnen System vereinigt werden. Hybride Empfehlungsansätze können darüber hinaus auch noch weitere Techniken enthalten, wie beispielsweise die Berücksichtigung von demografischen Daten oder bestimmten Eigenschaften eines Inhaltes [6].

Ein weiteres Gebiet sind risikobewusste Empfehlungssysteme, die berücksichtigen, dass bestimmte Empfehlungen für bestimmte Nutzer als störend einzustufen sind oder sogar dazu führen, Nutzer zu verlieren. Hierbei geht es darum, solche Fauxpas zu erkennen und auszufiltern [7].

II. KONZEPT

Als Anwendungsfall für den PoC soll ein Empfehlungssystem für Wikipedia realisiert werden. Dafür möchten wir für eine Untermenge von Wikipedia-Artikeln für jeden dieser Artikel Empfehlungen für andere Artikel von verschiedenen Empfehlungssystemen erhalten und damit die Empfehlungssysteme vergleichen. Wikipedia besteht aus Artikeln, die in Themengebiete aufgeteilt sind. Um die Qualität der Empfehlungen messen zu können, haben wir uns dafür entschieden, dass wir für einen Artikel als Empfehlung andere Artikel des gleichen Themengebietes erhalten möchten. Den Empfehlungssystemen soll die Themenzuordnung dabei nicht bekannt sein, sie sollen vielmehr die Zusammenhänge selbst entdecken. Dies spiegelt zwar nicht unbedingt die Realität wider, ist aber eine einfach und eindeutig zu bewertende Metrik.

Neben dem Wikipedia-Modell besteht unser PoC aus drei weiteren Komponenten, die im Folgenden näher beschrieben werden:

- Generierung der Trainingsdaten
- Empfehlungssystem
- Lokale Website

Der Gesamtaufbau ist schematisch in Abbildung 1 dargestellt.

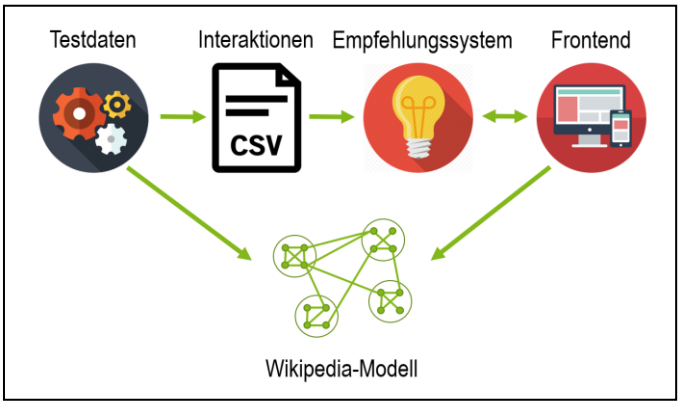

Abbildung 1. Schematischer Gesamtaufbau der Komponenten des PoC

A. *Wikipedia-Modell*

Für unseren PoC haben wir uns für ein manuell zusammengestelltes Wikipedia-Modell entschieden. Es besteht aus 79 Artikeln. Dabei wurden aus acht Themengebieten jeweils zehn Artikel (bzw. neun Artikel bei dem Themengebiet „Wissenschaft“) ausgewählt. Die Themengebiete entsprechen den acht „Themenportalen“ der deutschsprachigen Wikipedia. Diese 79 Artikel wurden logisch miteinander verbunden.

Sie bestehen dabei nur aus dem jeweiligen Titel, da die Inhalte der Artikel für den vorgesehenen Anwendungsfall nicht relevant sind. Die Verbindungen stellen die Querverweise innerhalb von realen Wikipedia-Artikeln dar. Sie sind unidirektional, da ein Artikel A auf den Artikel B verweisen kann, dieser aber nicht zwangsläufig wieder einen Verweis auf Artikel A enthalten muss. In einigen Fällen existieren in unserem Modell jedoch bidirektionale Verbindungen zwischen Artikeln, wenn sie thematisch eng miteinander verbunden sind (z.B. „Kirche“ und „Christentum“). Innerhalb der Themengebiete bestehen zwischen 20 und 29 Verbindungen der Artikel. Es wurde darauf geachtet, dass keine Artikel Sackgassen bilden und alle Artikel erreichbar sind. Zudem wurden 44 Verbindungen zwischen Artikeln verschiedener Themengebiete erstellt. Die Artikel sind daher in den meisten Fällen innerhalb ihrer Themengebiete enger miteinander verbunden als außerhalb.

Das Modell wurde in der Programmiersprache PHP beschrieben und kann durch ein selbst entwickeltes Skript formatiert im „JSON Graph Format“ (Version 2) ausgegeben werden. Dieses Format stellt jeden Artikel als Knoten dar, der einen Titel hat und einer Kategorie zugeordnet ist. Die Querverweise zwischen den Artikeln sind gerichtete Kanten zwischen den Knoten. Durch den Einsatz dieses festgelegten Formates lässt sich das Wikipedia-Modell als Graph durch verfügbare Software visualisieren (Abbildung 2).

B. *Website zur Navigation*

Unser PoC enthält außerdem eine lokale Website – ebenfalls in PHP entwickelt – welche die Navigation durch das Wikipedia-Modell auf Basis des Wikipedia-Modells der erzeugten JSON-Datei ermöglicht. Die Website listet auf der Startseite alle Artikel sortiert nach Themengebieten auf. Wählt man einen Artikel aus, werden alle Artikel angezeigt, die in unserem Modell direkt mit dem ausgewählten Artikel verbunden sind. Da der Graph zusammenhängend ist, lässt sich so jeder Artikel von jedem Artikel aus erreichen. Zusätzlich werden bei jedem Seitenaufruf Empfehlungen von

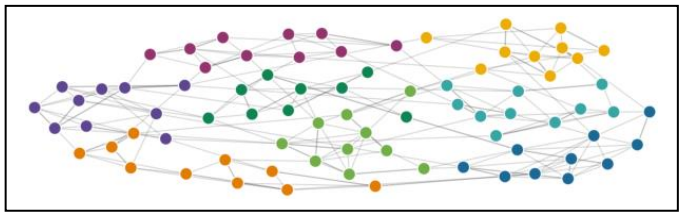

Abbildung 2. Darstellung des Wikipedia-Graphen, jede Farbe entspricht einem Themengebiet

den angebundenen Empfehlungssystemen angefordert und dargestellt. Außerdem wird der Verlauf der besuchten Artikel in Form einer Nutzersitzung gespeichert und bereits besuchte Artikel entsprechend markiert.

C. *Generierung von Trainingsdaten*

Die Grundlage für das Training des Empfehlungssystems sind Daten über die Interaktionen zwischen Benutzern und Artikeln. Da Wikipedia selbst solche Daten nicht erhebt bzw. veröffentlicht, generieren wir eigene Daten aus unserem Modell. Der Algorithmus dazu ist im Folgenden beschrieben.

Zuerst werden 1.000 Nutzer registriert. In einem Zeitstempel wird die aktuelle (simulierte) Zeit festgehalten. Für den ersten Nutzer wird eine Sitzung erstellt. Für die Sitzung wird festgelegt, welches Themengebiet der präferierte Bereich des Nutzers ist und wie viele Seitenaufrufe getätigt werden sollen. Es werden pro Sitzung zufällig zwischen 10 und 15 Aufrufe ausgeführt. Aus dem präferierten Themengebiet wird ein zufälliger Artikel als Startknoten ausgewählt. Von diesem Knoten ausgehend wird der Graph zufällig durchlaufen, bis die geforderte Anzahl erreicht wurde.

Dieser Vorgang wird für alle weiteren Nutzer wiederholt. Wenn alle Nutzer eine Sitzung hatten, wird wieder bei dem ersten Nutzer begonnen. Das präferierte Themengebiet kann für denselben Nutzer in unterschiedlichen Sitzungen durchaus verschieden sein.

Bei diesem Verfahren wird für jeden Seitenaufruf eine Datenreihe generiert. Diese besteht aus der Nutzer-ID, der Sitzungs-ID, dem aufgerufenen Artikel, dem aktuellen Zeitstempel und einer Bewertung. Der Zeitstempel wird nach jedem Aufruf um eine zufällige Dauer zwischen 5 und 20 Sekunden vorgerückt.

Die Bewertung eines Artikels errechnet sich aus der Distanz des Artikels zum präferierten Themengebiet der aktuellen Sitzung. Dabei entspricht die höchstmögliche Bewertung die „Breite“ des Graphen. In unserem Modell ist jeder Artikel von jedem anderen in maximal sieben Schritten erreichbar. Wenn für jeden Schritt, mit dem sich vom Themengebiet entfernt wird, die Bewertung um einen Punkt sinkt, kann die Bewertung also um maximal sieben Punkte fallen. Daher wird als Bewertungsschema eine Skala von null bis sieben gewählt.

Ein Artikel, der beispielsweise drei Schritte von der Zielkategorie entfernt ist, bekommt die Bewertung 7 – 3 = 4. Die Bewertungen werden anschließend in den Wertebereich [0.5, 5.0] normiert und dabei auf das nächste Vielfache von 0,5 gerundet. Damit soll ein 5-Sterne Bewertungssystem imitiert werden.

Der gesamte Vorgang wird solange wiederholt, bis die gewünschte Anzahl an Datenreihen generiert wurde. Der gesamte Datensatz wird in eine CSV-Datei geschrieben.

III. Amazon Personalize

Durch Amazon Personalize macht der Anbieter sein über Jahre angesammeltes Wissen über Empfehlungssysteme der Öffentlichkeit zugänglich. Es werden qualitativ hochwertige Empfehlungen in Echtzeit versprochen, die sich schnell und ohne technisches Wissen in Anwendungen und Dienste integrieren lassen sollen.

A. Erstellung eines Empfehlungssystems

Um Amazon Personalize verwenden zu können, muss zuerst ein Konto bei Amazon Web Services (AWS) angelegt werden. Mit diesem kann man sich in der AWS-Konsole anmelden, einem Webportal zur Interaktion mit AWS-Diensten. Zusätzlich wird ein Identity and Access Management (IAM)-Benutzer angelegt, der nur Zugriff auf Dienste von Amazon Personalize hat. Man kann sich das so vorstellen, dass der Hauptbenutzer den AWS-Kunden repräsentiert, also etwa ein Unternehmen. Dieser Benutzer hat vollen Zugriff auf alle Dienste und ist für die Zahlungen verantwortlich. Für einzelne Entwickler oder Applikationen können zugriffsbeschränkte IAM-Benutzer angelegt werden, die nur Zugang zu ihren benötigten Diensten haben.

In der AWS-Konsole legt man eine „dataset-group“ an, welche die Daten einer einzigen Applikation repräsentiert. Die dataset-group ist also wie ein Projekt zu sehen und wird im Folgenden auch so bezeichnet. Wenn man ein Projekt öffnet wird ein Dashboard präsentiert, über das der Workflow zur Erstellung eines Empfehlungssystems gesteuert wird. Der Workflow lässt sich in drei Schritte gliedern:

1) Importieren von Trainingsdaten
2) Trainieren eines Modells
3) Bereitstellung eines Dienstes mit dem Modell

Daten für den Algorithmus können entweder als „historische“ Daten in CSV-Form oder live als Event über einen REST-Endpunkt hochgeladen werden. In unserem PoC werden die generierten Trainingsdaten als CSV-Datei hochgeladen. Das geschieht über Amazon S3, ein Service für Online-Datenspeicher.

Um eine Datei hochladen zu können wird zuerst in S3 ein sogenanntes Bucket angelegt. Das geschieht ebenfalls über die AWS-Konsole. Zusätzlich müssen Zugriffsrechte auf den Bucket für Personalize gewährt werden.

Im Personalize-Dashboard können die Daten aus dem S3 Bucket importiert werden. Dazu muss in einem JSON-Format das Schema der zu importierenden CSV-Datei spezifiziert werden. Das ist im gesamten Workflow der einzige Code, der von Hand geschrieben werden muss.

Wenn die Interaktionen importiert wurden, kann ein Modell trainiert werden. Es stehen verschiedene Algorithmen für unterschiedliche Anwendungsfälle zur Auswahl. Für unseren PoC verwenden wir den „Item-based Collaborative Filtering“-Algorithmus, der für jeden Artikel verwandte Artikel berechnet. Er stellt somit eine Mischung aus den inhaltsbasierten und kollaborativen Ansätzen dar.

Nach Abschluss des Trainings kann das Empfehlungssystem veröffentlicht werden. Dazu muss nur spezifiziert werden, wie viele Transaktionen pro Sekunden (TPS) das System mindestens verarbeiten können muss. AWS reserviert die angeforderte Infrastruktur, die bei Bedarf auch selbstständig hochskaliert.

Um Empfehlungen für einen Artikel vom System anzufordern, muss nur eine REST-Anfrage an Personalize gestellt werden. Die Anfrage besteht aus dem Amazon-Ressourcennamen (ARN), dem Artikelnamen und der maximalen Anzahl der geforderten Empfehlungen. Der ARN ist die einzigartige Identifikation unseres Empfehlungssystems. Zur Authentifizierung wird die Anfrage zusätzlich signiert. Dazu wird in der AWS-Konsole für den oben genannten IAM-Benutzer ein Schlüsselpaar generiert. Dieser gesamte Ablauf kann mit dem AWS-SDK für PHP in unseren PoC integriert werden.

Damit sich das Empfehlungssystem über die Zeit anpassen kann, bietet Personalize eine Event API an. Dort können neue Nutzer und Artikel sowie die Interaktionen registriert werden. Das Empfehlungssystem wird regelmäßig neu trainiert, falls Events eintreffen.

B. Bewertung

Amazon Personalize bietet eine gesamtheitliche Lösung für Erstellung und Betrieb von Empfehlungssystemen, welche sich das angesammelte Know-How von Amazon zu Nutzen macht. Der gesamte Prozess ist ohne technisches Vorwissen und praktisch ohne Code in wenigen Stunden realisierbar. Dabei ist der Gesamtablauf bis ins Detail bestens dokumentiert. Auch die Integration in die Zielanwendung ist mithilfe der angebotenen SDKs für die gängigsten Programmiersprachen mit wenig Aufwand realisierbar.

Der größte Nachteil im Vergleich zu anderen verfügbaren Lösungen sind die Kosten (im Folgenden beziehen sich alle genannten Kosten auf den Stand vom Januar 2021). Für jeden hochgeladenen GB an Trainingsdaten, inklusive Event API, werden 0,05$ berechnet. Für das Training des Empfehlungssystems fallen für jede Stunde 0,24$ an. Eine Trainingsstunde bedeutet dabei eine Realstunde Rechenzeit auf vier virtuellen CPUs mit 8 GB Arbeitsspeicher. Es werden aber im Normalfall mehr Ressourcen verwendet, um die reale Trainingsdauer kurz zu halten. In einer Beispielrechnung von Amazon Personalize wird angegeben, dass zehn Trainingsstunden innerhalb von 20 Minuten abgearbeitet werden. Das entspricht der Parallelisierung von 30 Instanzen. Die Bereitstellung des Empfehlungssystems wird in TPSh berechnet. Eine Infrastruktur verbraucht eine TPSh, wenn sie eine Stunde bereitgestellt wird und dabei eine Transaktion pro Sekunde (TPS) verarbeiten kann. Bei einer Benutzung von bis zu 20.000 TPSh pro Monat werden 0,20$ pro TPSh berechnet, weitere 180.000 TPSh kosten jeweils 0,10$ und alles darüber noch 0,05$. Diese Kosten sind entsprechend höher, wenn das System überlastet wird und die Infrastruktur automatisch hochskaliert wird. Werden in einem Monat beispielsweise 25.000 TPSh konsumiert, dann fallen insgesamt 20.000 * 0,20$ + 5.000 * 0,10$ = 4.500$ an.

IV. LibRec

LibRec ist eine Java-Bibliothek unter GPLv3-Lizenz, die nach eigenen Angaben über 70 verschiedene Algorithmen aus dem Bereich Empfehlungssysteme enthält, wodurch sich viele Anwendungsszenarien abdecken lassen. Die Bibliothek ist seit Januar 2014 auf GitHub verfügbar, seit 2018 hat die Aktivität des Repository jedoch stark nachgelassen. Die Bibliothek wird zwar nur von zwei Personen betreut, es existieren aber mindestens 1.000 Forks des Repository, wodurch auszugehen ist, dass das Projekt durch andere Entwickler weiterhin fortgeführt wird.

LibRec unterstützt die vorgestellten Prinzipien von inhaltsbasierten und kollaborativen Empfehlungen und erlaubt darüber hinaus auch die Kombination der Ansätze (vgl. Kapitel I.B).

A. Aufbau der Bibliothek

Die Bibliothek enthält bestimmte universelle Komponenten, die durch verschiedene Implementierungen und Algorithmen passend zum gewünschten Verhalten miteinander kombiniert werden können. Außerdem lassen sich durch diesen Aufbau eigene Implementierungen der Komponenten erstellen und verwenden.

Eingabedaten, so beispielsweise Trainingsdaten, werden durch die Komponente „DataModel“ eingelesen und verwaltet. Diese können als Textdateien in verschiedenen Formaten vorliegen. Außerdem sind andere Datenquellen möglich, wie z.B. Datenbanken über JDBC. Die Komponente „RecommenderContext“ wird verwendet, um mit den Eingabedaten durch Algorithmen im Bereich des maschinellen Lernen ein Modell für die weitere Verarbeitung zu generieren. Hierfür wird unter anderem ein Algorithmus angeboten, der eine Ähnlichkeitsmatrix nach Korrelationskoeffizient aufbaut. Dieser wurde in unserem PoC verwendet, um Nutzer nach ihren Bewertungen zu gruppieren. „Recommender“ bildet die Komponente, die dann Empfehlungen auf Basis des Modells generiert. In dem vorliegenden Fall basiert der eingesetzte Algorithmus auf der Nächste-Nachbarn-Klassifikation („k-nearest neighbors algorithm“). Dadurch werden die einem bestimmten Nutzer ähnlichsten Nutzer innerhalb der Matrix gefunden. Eine weitere Komponente bildet der „RecommenderEvaluator“, der berechnet, wie gut die generierten Empfehlungen sind. Hierfür ist beispielsweise eine Implementierung zur Berechnung des mittleren absoluten Fehlers verfügbar.

B. Einsatz der Bibliothek im Rahmen des PoC

Mithilfe der Bibliothek LibRec wurde eine prototypenhafte Anwendung entwickelt, die bei Anwendungsstart die Trainingsdaten einliest, die Ähnlichkeitsmatrix aufbaut und diverse weitere Schritte zur Optimierung und Weiterentwicklung des Modells durchläuft. Anschließend wird das Modell evaluiert, wobei es bei größeren Abweichungen verworfen und neu erzeugt wird.

Das Erzeugen und Trainieren des Modells führt bei gleichen Trainingsdaten bei jeder Ausführung zu anderen Ergebnissen. Es ist daher nicht reproduzierbar und die verwendeten Algorithmen arbeiten vermutlich zu einem gewissen Teil mit Zufallswerten. Das Ergebnis ist somit jedes Mal ein anderes Modell, welches zu anderen Empfehlungen führt. Leider bietet die Bibliothek keine Implementierung zum Speichern und Laden der erzeugten Modelle. Nach Durchsicht des Quellcodes scheint es aber mit vertretbarem Aufwand möglich zu sein, die Bibliothek um diese Funktionalitäten zu erweitern.

Um den Prototypen über eine REST-Schnittstelle an unsere Website anbinden zu können, wurde die Anwendung mithilfe des Frameworks Spring Boot entwickelt. Es wurde ein POST-Endpunkt integriert, der eine Liste von empfohlenen Artikeln basierend auf einem als Parameter übergebenen Artikel zurückgibt. Anfrage- und Antwortdaten sind dabei JSON-formatiert.

Sobald die Trainingsschritte abgeschlossen sind, wird der Endpunkt bereitgestellt und es lassen sich Empfehlungen generieren. Der Empfehlungsprozess für verwandte Artikel wird im Folgenden vorgestellt.

C. Generierung von Empfehlungen zu einem Artikel

Um Empfehlungen zu einem Artikel, beispielsweise „Medizin“ aus dem Themengebiet „Wissenschaft“, zu generieren, werden die folgenden Schritte durchgeführt. Zuerst wird abgefragt, welche Nutzer den Artikel „Medizin“ gut bewertet haben. In den simulierten Interaktionen der Nutzer haben die Nutzer Artikel innerhalb ihres präferierten Themengebietes mit voller Punktzahl bewertet und Artikel außerhalb schlechter. Das generierte Modell von LibRec gruppiert die Nutzer nach Artikeln, die sie gut bewertet haben. Im weiteren Sinne werden die Nutzer daher nach den Themengebieten des Wikipedia-Modells gruppiert. Wird also untersucht, welche Nutzer den Artikel „Medizin“ hoch bewertet haben, werden hauptsächlich Nutzer gefunden, deren Präferenz das Themengebiet „Wissenschaft“ ist.

Als nächstes wird für diese Nutzer abgefragt, welche Artikel von diesen Nutzern gut bewertet wurden. Das Ergebnis ist eine Liste von Artikeln und den dazugehörigen Bewertungen. Es ist davon auszugehen, dass diese Liste hauptsächlich Artikel aus dem Themengebiet „Wissenschaft“ enthält.

Als letzten Schritt muss die Liste noch aufbereitet werden. Da mehrere Nutzer gleiche Artikel unterschiedlich bewertet haben, enthält die Liste viele Dubletten. Für jeden mehrfach vorkommenden Artikel wird die höchste vergebene Bewertung verwendet. Die restlichen Bewertungen werden verworfen. Der ursprüngliche Artikel („Medizin“) ist höchstwahrscheinlich auch in der Liste enthalten und wird entfernt, da er als Empfehlung keinen Sinn ergibt. Abschließend werden die Artikel absteigend nach ihren Bewertungen sortiert, sodass die Ergebnisliste mit den besten Empfehlungen beginnt.

D. Mögliche Verbesserungen der Empfehlungen

Wie bereits erwähnt basieren die Empfehlungen nur auf dem einen Artikel, der dem Endpunkt übergeben wird. Dabei wird der Verlauf des Nutzers nicht berücksichtigt. Doch gerade das aktive Berücksichtigen der Interessen des Nutzers während einer Sitzung sowie das Erzeugen eines präzisen Profils machen Empfehlungssysteme so erfolgreich. Eine logische Erweiterung des PoC wäre daher, Empfehlungen anhand bisheriger Präferenzen und nicht nur auf Basis eines Artikels zu erzeugen. Dafür wäre denkbar, auf der Website nicht nur den Verlauf zu erfassen, sondern diesen auch zu bewerten. Die Bewertung wäre durch eine Zeiterfassung möglich, wie lange ein Nutzer auf einem bestimmten Artikel bleibt. Eine andere Möglichkeit stellen aktive Bewertungen dar, beispielsweise in Form von Sterne-Bewertungen oder „Gefällt mir“-Angaben, die ein Nutzer vergeben kann. Ist nach einiger Zeit ein bewerteter Verlauf verfügbar, könnte er in Verbindung mit dem aktuellen Artikel als Basis für Empfehlungen verwendet werden. Dabei könnten sehr gut bewertete Artikel zu genaueren Empfehlungen über ein breiteres Spektrum führen. Schlecht bewertete Artikel könnten zudem als Ausschlusskriterium für ähnliche Artikel verwendet werden.

V. Ergebnisse

A. Methodik

Um die Empfehlungssysteme automatisiert testen zu können, wurde ein PHP-Skript entwickelt, welches für alle 79

Artikel des Wikipedia-Modells von beiden Empfehlungssystemen nacheinander jeweils fünf Empfehlungen anfordert. Dabei wird gezählt, wie viele Artikel aus den Empfehlungen aus demselben Themengebiet stammen. Dies wird zur Messung der Genauigkeit der Empfehlungen genommen. Sobald ein Artikel aus einem anderen Themengebiet stammt, wird er als Abweichung geloggt. Zudem wird geloggt, wenn bei einer Empfehlung weniger als die geforderten fünf Artikel angeboten werden.

B. Ergebnisse und Bewertung

Amazon Personalize bietet nahezu perfekte Ergebnisse. Über 95% der Empfehlungen sind Artikel aus dem jeweils richtigen Themengebiet. Dazu sind die Abweichungen in den meisten Fällen nachvollziehbar, da sie auftreten, wenn Artikel themenübergreifend stärker miteinander verbunden sind als sie es innerhalb ihrer Themengebiete sind. So schlägt Amazon Personalize beispielsweise für den Artikel „Gebirge“ aus dem Themengebiet „Geographie“ den Artikel „Wintersport“ aus dem Bereich „Sport“ vor. Diese Artikel sind durch die 44 Verbindungen von Artikeln aus verschiedenen Themengebieten bidirektional direkt miteinander verbunden. Abweichungen wie diese sollten daher nicht als Fehler gewertet werden.

Die Lösung mit LibRec liefert je nach Modell eine Genauigkeit zwischen 64% und 73%. Zudem sind die Ergebnisse deutlich stärker von den Trainingsdaten abhängig als bei Amazon Personalize. Die Anzahl der verschiedenen Nutzer, die Anzahl ihrer Bewertungen sowie die Gesamtanzahl von Interaktionen wirken sich enorm auf die Ergebnisse aus, die in einigen Fällen lediglich eine Genauigkeit von 9% bis 15% liefern, was bei acht Themengebieten im Bereich des Zufalls liegt. Werden nicht die jeweils besten Bewertungen jedes Artikels verwendet, sondern der Durchschnitt daraus berechnet (vgl. Kapitel IV.C), so fällt die Qualität der Ergebnisse mit denselben Trainingsdaten um über 10%.

Neben der Qualität ist auch die Geschwindigkeit eine sinnvolle Metrik. Die durchschnittliche Antwortzeit von Amazon Personalize ist etwa sechs- bis zehnmal länger als die der lokalen LibRec/Spring Boot-Anwendung. Unser Amazon Personalize-Deployment wird im AWS-Rechenzentrum in Frankfurt am Main betrieben. Dadurch wird im Gegensatz zur lokalen Anwendung ein vergleichsweise weiter Weg über öffentliche Netze zurückgelegt, der Zeit kostet. Möchte man seinen selbst gehosteten Online-Dienst um ein Empfehlungssystem erweitern, ergibt sich mit Lösungen wie LibRec die Möglichkeit, das Empfehlungssystem auf dem gleichen Server zu betreiben. Externe Dienste wie Amazon Personalize können dagegen grundsätzlich nicht lokal betrieben werden. Falls der gesamte Online-Dienst bei dem gleichen Betreiber gehostet wird, wird dieser Latenzunterschied eventuell reduziert. Dieser Fall wurde nicht weiter untersucht.

Die Antwortzeit auf eine einzige Empfehlungsanfrage liegt bei Amazon Personalize bei etwa 0,22 Sekunden und bei der LibRec/Spring Boot-Anwendung bei 0,04 Sekunden. Werden in dem bereits beschriebenen Testlauf zu allen 79 Artikeln einzeln Empfehlungen angefordert, verhält sich der Zeitunterschied gleich.

Eine weitere Auffälligkeit besteht darin, dass bei steigender Anzahl von Trainingsdaten nicht nur die Trainingsdauer des Modells von LibRec zunimmt, sondern auch die Generierungszeit von Empfehlungen. Bei 10.000 Interaktionen als Trainingsgrundlage werden die oben genannten Zeiten erreicht. Werden dagegen 100.000 Interaktionen verwendet, so werden durchschnittlich 0,11 Sekunden pro Empfehlungsanfrage benötigt. Zusätzlich wird die Qualität der Ergebnisse unerwartet schlechter. Mehr Trainingsdaten führen bei der LibRec-Variante also nicht zu einem präziseren Modell, sondern zu einem deutlich größeren. Aus diesem Grund wurde die Variante mit LibRec mit 10.000 Datensätzen betrieben. Amazon Personalize wurde dagegen mit 100.000 Datensätzen trainiert, da hier keine Leistungseinbußen festgestellt werden konnten.

VI. Weitere Systeme

Für AWS existieren einige Konkurrenzdienste wie beispielsweise die Cloud Computing-Plattform Microsoft Azure. Diese bietet nicht direkt eine vergleichbare Alternative zu Amazon Personalize, bietet aber immerhin einige Leitfäden, wie Empfehlungssysteme allgemein [8] sowie speziell unter Verwendung von Microsoft Azure entwickelt und betrieben werden können. Zahlreiche Informationen und Beispiele dazu finden sich unter anderem in der Azure-Dokumentation oder in den GitHub-Repositories von Azure.

Auch für selbst betriebene Lösungen sind mehrere Alternativen verfügbar. So bietet beispielsweise die Machine Learning Library (MLlib) von Apache Spark eine Unterstützung für Collaborative Filtering. Ebenfalls interessant ist TensorFlow, eine Plattform für maschinelles Lernen, welche unter anderem von Google aktiv entwickelt wird. Die Plattform erschien bereits im Jahr 2015, die Python-Bibliothek „TensorFlow Recommenders“, speziell für Empfehlungssysteme, ist dagegen erst seit September 2020 verfügbar. Daneben gibt es Bibliotheken anderer Entwickler für Empfehlungssysteme, die auf TensorFlow basieren, wie beispielsweise die Python-Bibliothek „LibRecommender“. Zudem ist mit „librec-auto“ ein Python-Wrapper für die Java-basierte LibRec-Bibliothek verfügbar.

Zusammenfassend kann festgehalten werden, dass mehrere Bibliotheken von unterschiedlichen Anbietern für unterschiedliche Ökosysteme verfügbar sind, mit denen man Empfehlungssysteme aufbauen kann. Bevor man ein solches System für den Produktiveinsatz umsetzt, sollte eine umfangreiche Recherche und Evaluierung durchgeführt werden. Dabei sollten neben Kosten, Verfügbarkeit, Funktionsumfang, Leistung und Skalierbarkeit auch die Dokumentation und Aktivität der Bibliotheken und externen Dienstleistungen berücksichtigt werden.

VII. Fazit

Empfehlungssysteme sind aus dem modernen Internet nicht mehr wegzudenken. Bei Unternehmen wie Netflix oder Amazon werden durch Empfehlungen enorme Summen an Geschäftswert generiert [1, 2]. Dem Nutzer wird auf der Plattform dadurch ein personalisiertes Erlebnis geboten.

Für die Realisierung eines eigenen Empfehlungssystems gibt es eine diverse Landschaft von verfügbaren Lösungen und Algorithmen. Viele davon sind frei zugänglich, einige nur kommerziell. In dieser Arbeit wurden die kostenpflichtige Gesamtlösung Amazon Personalize und die frei verfügbare Bibliothek LibRec genauer untersucht.

Für Amazon Personalize sprechen die präzisen und sinnvollen Empfehlungen, die mit sehr wenig Eigenaufwand bis in eine Produktionsumgebung ausgerollt werden können.

Dieser hohe Grad an Autonomie auf Seiten von Amazon kommt mit entsprechenden finanziellen Kosten, weshalb diese Lösung beispielsweise für Startups und Anbieter mit überschaubaren Produktpaletten nicht unbedingt einen wirtschaftlichen Mehrwert bietet. Amazon Personalize ist durch die verfügbaren Ressourcen von AWS und vermutlich besser optimierten Algorithmen fast beliebig skalierbar und wir haben mit unserem PoC sicherlich nur am unteren Ende des Machbaren gekratzt.

Durch die Implementierung eines eigenen Empfehlungssystems mithilfe von LibRec konnten wir zeigen, dass auch mit angemessenem Aufwand akzeptable Empfehlungen erzielt werden können. Auch andere verfügbare Lösungen wie Spark MLlib und TensorFlow Recommenders bieten vielversprechende Ansätze.

LITERATUR

[1] A. Sharma, J.M. Hofman, D.J. Watts, "Estimating the Causal Impact of Recommendation Systems from Observational Data", Proc. 16th ACM Conf. Economics and Computation, 2015, pp. 453-470

[2] C.A. Gomez-Uribe and N. Hunt, "The Netflix Recommender System: Algorithms, Business Value, and Innovation", ACM Trans. Management Information Systems, vol. 6, no. 4, 2016, pp. 1–19

[3] G. Linden, B. Smith and J. York, "Amazon.com recommendations: item-to-item collaborative filtering", IEEE Internet Computing, vol. 7, no. 1, 2003, pp. 76-80

[4] B. Smith and G. Linden, "Two Decades of Recommender Systems at Amazon.com", IEEE Internet Computing, vol. 21, no. 3, 2017, pp. 12-18

[5] R. Meteren, M. Someren, "Using Content-Based Filtering for Recommendation", 2000, pp. 2-3

[6] R. Burke, "Hybrid Recommender Systems: Survey and Experiments", User Modelingand User-Adapted Interaction 12, 2002, pp. 331-335

[7] D. Bouneffouf, A. Bouzeghoub, A.L. Ganarski, "Risk-Aware Recommender Systems", International Conference on Neural Information Processing, 2013, p. 1

[8] A. Argyriou, M. González-Fierro, L. Zhang, "Microsoft Recommenders: Best Practices for Production-Ready Recommendation Systems", Companion Proceedings of the Web Conference 2020, pp. 50-51