K-Nearest Neighbors (KNN) in Empfehlungssystemen - eine Literaturrecherche

Jan Arends
Fachbereich Informatik
Hochschule Bonn-Rhein-Sieg
Sankt Augustin, Germany
jan.arends@smail.inf.h-brs.de

***Abstract*—Diese Arbeit beschäftigt sich mit der Fragestellung, in wie weit sich der k-nrearest Neighbors Algorithmus (kNN) in Empfehlungssystemen (eng. Recommendation Engines) einsetzen lässt. Die Arbeit soll somit die Relevanz von kNN in Empfehlungssystemen darlegen. Dabei steht die praktische Anwendbarkeit des Algorithmus in diesem Kontext im Vordergrund. Das beinhaltet eine Untersuchung dahingehend, wie ausgereift das Verfahren bei Empfehlungssystemen ist, ob und wie es bereits in der Praxis eingesetzt werden kann und welche Probleme oder Fragestellungen derzeit in der Forschung und in der Praxis offen sind.**

***Index Terms*—kNN, Recommendation Engine, collaborative filtering, content-based-filtering, item-based-filtering**

I. Einleitung

Empfehlungssysteme sind Software-Tools und Techniken, die einem bestimmten Benutzer Vorschläge für bestimmte Objekte aus dem System, wie beispielsweise Artikel oder Filme, bieten, welche für diesen Benutzer am ehesten von Interesse sind [1]. Die Objekte werden in dieser Arbeit Items genannt.

Empfehlungssysteme unterstützten somit ihre Benutzer bei verschiedenen Entscheidungsprozessen anhand dessen Vorschläge. Um welche Art von Items es sich handelt kann sehr vielseitig sein. Aufgrund der Filter Techniken, die im laufe der Arbeit vorgestellt werden, und den allgemeinen Techniken der Informatik sind hier kaum Grenzen gesetzt. So kann ein Empfehlungssystem beispielsweise bei der Frage unterstützen, welche Artikel ein Nutzer kaufen, welche Musik der Nutzer hören oder welche Nachrichten er/sie lesen wollen würde. Empfehlungssysteme sind heutzutage ein wertvolles Mittel für Online-Benutzer, um die Informationsflut, welche Sie geboten bekommen, zu bewältigen und ihnen zu helfen, bessere Entscheidungen zu treffen. Sie sind heute eines der leistungsfähigsten und beliebtesten Tools zur Informationsfindung im Internet.

Es wurden mehrere Techniken zur Empfehlungsgenerierung publiziert und in den letzten zehn Jahren wurden viele von ihnen erfolgreich in kommerziellen Umgebungen eingesetzt [1]. Tatsächlich zielt die Forschung zu Empfehlungssystemen im Allgemeinen, abgesehen von ihrem theoretischen Beiträgen, auf die praktische Verbesserung industrieller Empfehlungssysteme ab und beinhaltet verschiedene praktische Aspekte, die sich auf die Implementierung der Systeme beziehen. Empfehlungssysteme sind ein Beispiel für den großflächigen Einsatz von Machine Learning und Data-Mining-Algorithmen in der kommerziellen Praxis [1].

Eines der einfachsten und leicht verständlichen Verfahren im Bereich Maschine Learning ist der k-nearest Neighbor Algorithmus, kurz kNN. Trotz seines simplen Vorgehens ist der Algorithmus sehr effektiv. Durch diese Kombination an Eigenschaften ist kNN so beliebt und entsprechend weit verbreitet. Mit ”nearest” in dem Namen des Algorithmus ist hier die Entfernung verschiedener Punkten in einem n-Dimensionen Raum gemeint. Je geringer die Entfernung der Objekte in diesem Raum, je ähnlicher sind sich diese Objekte. Mit dieser Denkweise kann man mittels des kNN Verfahrens Klassifizierungs- und Regressionsprobleme lösen.

Es lässt sich rein Intuitiv einen Zusammenhang zwischen kNN und Empfehlungssystemen erkennen. Finden man nämlich Items welche ähnlich zueinander sind und das System weiß, dass einem Nutzer eines dieser Items gefällt, ist die Wahrscheinlichkeit hoch, dass dem Nutzer auch ähnliche Items gefällt könnten. Wie man ähnliche Items ausfindig macht und vorhersagt, welche Items dem Benutzer gefallen könnten, wird in dieser Arbeit vorgestellt. Zunächst jedoch wird das kNN Verfahren sowie Recommendaion Engines im Allgemeinen vorgestellt.

A. K-Nearest Neighbors

Der kNN Algorithmus ist eine sogenannte *Supervised Learning* Methode. Das bedeutet, dass die einzelnen Daten aus dem Datensatz bereits mit einem Label versehen sind, welches die einzelnen Elemente mit einer Klasse kennzeichnen. Der Algorithmus bildet dementsprechend nicht selber die Klassen, wie es *Unsupervised Learning* Methoden tun.

Der kNN benötigt keine Lernphase sondern berechnet die nächstgelegenen Nachbarn *direkt* und muss dazu den ganze Datensatz im Arbeitsspeicher halten. Der Algorithmus wird daher auch als *layz learner* bezeichnet. Die Tatsache dass

kNN das komplette Datensatz im Arbeitsspeicher halten muss wird derm Algorithmus grundsätzlich als Schwäche ausgelegt, wenn es um große Datenmengen geht. Es bietet aber gleichzeitig den Vorteil, dass kein angelerntes Model entworfen und gewartet werden muss, was kNN von Design aus sehr agil macht.

Der kNN Algorithmus ist mit einer der gängigsten Algorithmen im Bereich Empfehlungssystemen. In der Tat findet man den Algorithmus überall in der Literatur im Kontext von Empfehlungssystemen. Grund dafür ist, dass das Grundkonzept sich sehr gut auf das Empfehlungsproblem abbilden lässt. Items zu finden, welche ähnlich sind zu den Items sind, die ein Nutzer mag ist im wesentlichen genau das, was kNN liefert [2].

Typischerweise wird kNN für Klassifizierungen benutzt, kann jedoch auch für Regression benutzt werden. Führen wir zunächst die Klassifizierung mitteln kNN ein. Gegeben sei ein Punkt in einem n-Dimensionalen Raum, welche durch einen Vektor beschrieben ist und Klassifiziert werden soll. kNN berechnet zunächst die Entfernungen zu jedem Punkt aus dem Datensatz anhand einer Ähnlichkeitsfunktion, auch genannt *distance metric*, auf die wir später noch genauer zu sprechen kommen. Anschließend filtert der Algorithmus die Punkte anhand des Ergebnis der Ähnlichkeitsfunktion, sodass nur noch die k Nachbarn verbleiben, welche dem neuen Punkt am naheliegensten sind - die k-nearest Neighbors. Mithilfe dieser Nachbarn bestimmt der Algorithmus die Klasse des neuen Punktes. Die Grundlegende Idee dabei ist, dass wenn ein unklassifizierter Punkt in einer bestimmten Nachbarschaft fällt, dass dieser dann die Klasse annimmt, welche dominiered ist hinsicherlicht seiner Nachbarn.

In Bild 1 wird das Verfahren in einem zwei Dimensionalen Raum illustriert. In dem Beispiel gilt es den neuen Punkt, das Dreieck, welches mit einem Fragezeichen gekennzeichnet ist, in eines der beides Klassen zuzuordnen, also entweder zu der Klasse der Quadrate oder den Kreisen. Das Diagramm zeigt im linken Teil den Punkt, welchen es zu klassifizieren gilt, und im rechten Teil die k-nearest Neighbors mit k=1 und k=7. Bei k=1 würde der das Dreieck der Klasse der Quadrate zugeordnet werden, bei k=7 würde der Punkt allerdings der Klasse der Kreise zugeordnet werden. Es verdeutlicht somit auch, dass die Wahl von k die Klassifizierung beeinflussen kann. Der Wahl von k liegen viele Faktoren zu Grunde, welche hier nicht im Detail erläutert werden.

Wie wir später sehen werden, können die Bewertungen von Nutzern die Klassen in einem Empfehlungssystem darstellen. Diese Bewertung gilt es dann vorherzusagen. Je nach Bewertungsschema ist eine Regression angemessen, welches mit dem kNN ebenfalls berechnet werden kann. Der wesentliche Unterschied zwischen kNN Regression und kNN Klassifizierung ist, dass bei der Regression der Output keine Klasse ist, sondern ein Zahlenwert welcher den Durchschnitt der Nachbarn darstellt.

B. Recommendation Engines

Nachdem der kNN Algorithmus vorgestellt wurde, folgt nun eine Einführung in Empfehlungssysteme im Allgemeinen. Es gibt viele Gründe warum Service Anbieter die Funktionen eines Empfehlungssystems einsetzen wollen. Anschließend sind einige beispielhafte Motivationensgründe aufgelistet [1].

- Anzahl der verkauften Artikel erhöhen
- Die Diversität von verkauften Artikeln erhöhen
- Kundenzufriedenheit erhöhen
- Benutzerfreundlichkeit erhöhen
- Benutzer besser verstehen

Neben den Motivationen der Service Anbieter haben Service Nutzer ebenfalls ein Interesse an Empfehlungssystemen, sofern sie dem Nutzer bei der Entscheidungsfindung unterstützen. Hier aus Ausschnitt [1].

- Gute oder bewährte Items finden
- Ganze Serien oder Gruppen von Items finden
- Das Stöbern (also ohne Ziel etwas zu kaufen oder konsumieren) erleichtern
- Profil ergänzen mit Sachen die der User mag oder nicht mag

Damit ein Empfehlungssystem funktioniert, muss das System wissen, was ein Nutzer mag. Es ist also notwendig die Präferenzen jedes einzelnen Nutzers so gut es geht zu kennen, um somit die Menge an Items, welche dem Nutzer angezeigt werden, auf die vielversprechendsten Items zu reduzieren. Im allgemeinen heißt es daher auch, dass Empfehlungssysteme Item für den Benutzer *filtern*. Auf diesem Begriff in Kapitel I-B2 näher eingegangen. Zunächst wird jedoch vorgestellt, wie es Möglich wird die Präferenzen der Nutzer zu erfassen. Denn das ist essenziell, um das Empfehlungsproblem zu lösen, welches in [3] wie folgt definiert wurde.

> The recommendation problem can be defined as estimating the response of a user for new items, based on historical information stored in the system, and suggesting to this user novel and original items for which the predicted response is high.

1) Nutzer kennen lernen: Mit "user-item reposes" aus der obrigen Definition sind schlussendlich Bewertungen gemeint, die der Nutzer für Items aus dem System abgibt. Jegliche Art von User-Item Response wird daher im weiteren Verlauf der Arbeit *Bewertung* oder *Rating* genannt. Bewertungen können mittels zahlreicher Methoden gesammelt werden. Am praktischsten und hochwertigsten ist explizites Feedback, bei dem die Nutzer direkt über ihr Interesse an Produkten (oder allgemein Items) berichten. Beispiele dafür sind Bewertung mittels eines Sterne-System bei dem ein Stern unzufrieden bedeutet und 5 Sterne vollste Zufriedenheit, oder binäre Werte wie ein Likes oder Dislikes. Da explizites Feedback nicht immer verfügbar ist, leiten einige Empfehlungssysteme die Präferenzen der Benutzer aus impliziten Feedback ab, welches in der Regel reichlich vorhandenen ist. Durch Beobachtung des Benutzerverhaltens kann man so indirekt die Meinung des Benutzers implizieren, z.B. anhand der Bestellhisorie der

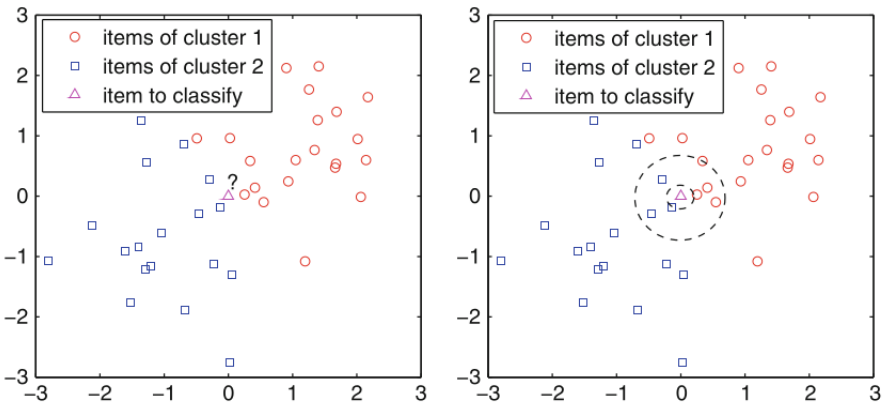

Fig. 1. Beispiel kNN Verfahren [2]

Nutzer oder Access Pattern [3] [4].

2) Techniken: Dadurch das Empfehlungssysteme heutzutage eine so wichtige Rolle spielen, gibt es entsprechend viele Paper die dieses Thema behandeln. So wurde in den letzten 20 Jahren eine Vielzahl von Techniken publiziert. Oft handelt es sich dabei um Erweiterungen oder Varianten von bestehenden Techniken. In diesem Abschnitt wird ein Überblick der Techniken gegeben und erklärt, bei welcher sich kNN vor allem eignet.

Empfehlungstechniken können anhand ihrer Wissensquellen unterschieden werden. In einigen Systemen ist dieses Wissen die Präferenzen der Benutzer, was in dieser Arbeit im Vordergrund steht. In anderen ist es ontologisches oder inferentielles Wissen über die Domäne, das von einem menschlichen Wissensspezialisten hinzugefügt wird [5].

Diese Unterscheidungen wurden bereits in verschiedenen Studien identifiziert. Eine umfangreiche Literatur Recherche erschien zuletzt 2012 in dieser Arbeit [6]. Bild 2 gibt eine erste Übersicht der verschiedenen Klassen mit ihren zugehörigen Wissensquellen.

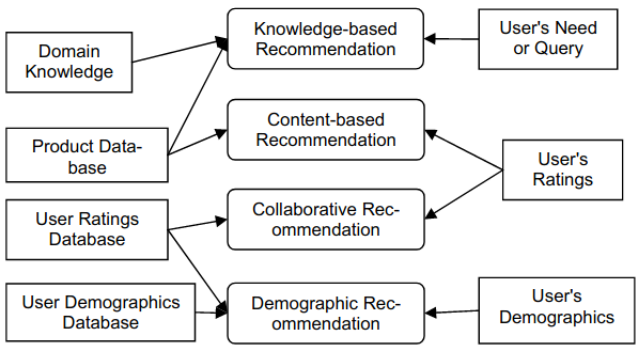

Fig. 2. Recommendation Techniken mit seinen Wissensquellen [5]

Bereits in 2002 indentifizierte man zwei wesentliche Typen von Empfehlungssystemen [7].

a) content-based filtering: Inhalts-basierte Empfehlungssysteme versuchen, Artikel zu empfehlen, die denen ähnlich sind, welchen ein bestimmter Benutzer in der Vergangenheit positiv bewertet hat. Der grundlegende Prozess, der von einem Inhalts-basierten Empfehlungssystem durchgeführt wird, besteht darin, die Attribute eines Benutzerprofils, in dem Vorlieben und Interessen gespeichert sind, mit den Attributen eines Inhaltsobjekts abzugleichen, um dem Benutzer neue ähnliche Artikel zu empfehlen [1]. Die diesem Sinne sind ähnliche Artikel also benachbart. Diese Art Empfehlungen zu machen war eines der ersten Ansätze.

b) collaborative filtering: Das kollaborative Filtern hingegen nutzt andere Informationen als Grundlage um Empfehlungen zu machen. Anstatt sich auf konkrete Produkteigenschaften zu fokussieren, beruhen kollaborative Filtermethoden auf Bewertungen von anderen Nutzern der Items in einem System. Die Grundidee besteht darin, dass das Rating eines Nutzers, für welchen es ein neues Produkt zu evaluieren gilt, wahrscheinlich ähnlich ist zu dem Rating von anderen Benutzern, sofern beide Benutzer in der Vergangenheit andere Produkte ähnlich bewertet haben [3]. Dementsprechend ist die Wahrscheinlichkeit hoch, dass eine in Frage stehende Bewertung ähnlich zu der eines anderen Nutzer ist, sofern sich diese Nutzer ähnlich sind. Ähnlichkeit bedeutet hier, dass die Präferenzen der Nutzer nahe beieinander liegen. Sind die Benutzer sich ähnlich, nennt man diese Nachbarn. Bei kollaborativen Filtern kann das Augenmerk aber auch auf den Items liegen anstatt auf den Benutzern. Das wird in Kapitel II weiter diskutiert.

Zusätzlich wurden in dieser Arbeit [5] aus 2007 die folgenden Klassen identifiziert (angelehnt an [8]): Demographic & Knowledge-Based. Diese sind für diese Arbeit jedoch nicht relevant, weswegen auf eine nähre Erläuterung verzichtet wird. Insgesamt gelten die vier Techniken seither als eine klassischen Unterteilung von Empfehlungssystemen [1]. Eine weitere, aktuelle Studie bestätigt, dass die gerade genannte

Unterteilung immer noch eine gängig ist, denn diese werden hier [9] ebenfalls identifiziert.

Die oben genannten Studien identifizieren zudem hybride Ansätze, welche die oben genannten Techniken kombinieren. Ein hybrides System, das die Techniken A und B kombiniert, versucht die Vorteile von A zu nutzen, um die Nachteile von B zu fixieren [1]. In [5] werden ca. 40 hybride Ansätze vorgestellt, welche auf den oben genannten Basis-Techniken basieren. In der Praxis sind hybride Ansätze gängig. Dieser Arbeit beschäftigt sich jedoch mit den Basis-Techniken und wie diese konkret mit kNN implementiert werden.

C. Abgrenzung

Der kNN Algorightmus kommt vor allem bei kollaborativen Filter Methoden zum Einsatz. Daher gehen wir hier näher auf dieser ein. Die Filtermethode lässt sich zunächst in zwei verschiedene Ansätze unterteilen:

- Nachbarschafts-basiert (neighboorhood-based) und
- Modell-basiert (modell-based).

Die beiden Methoden unterscheiden sich daran, ob die Empfehlung *direkt* oder *indirekt* aus den Daten erfolgt. Der Nachbarschaft-basierte Ansatz leitet Empfehlungen *direkt* anhand der Daten her. Im Gegensatz dazu wird beim Modell-basierten Ansatz zunächst ein Vorhersagemodell (predictive model) anhand der Daten entworfen, welches trainiert werden muss, um die Bewertung vorherzusagen und Empfehlungen machen zu können [3]. Hier kommen die Daten also *indirekt* zum Einsatz. Dieser Ansatz wurde hier nur der Vollständigkeit halber erwähnt. Diese Arbeit legt den Fokus auf Nachbarschaft-basiert Ansatz. Dieser lässt sich wiederum in zwei Kategorien unterteilen: *User-based* und *Item-based*. Das Schaubild gibt einen Überblick der gerade eingeführten Begriffe.

Nachbarschaft-basierte Ansätze bieten einige Vorteile

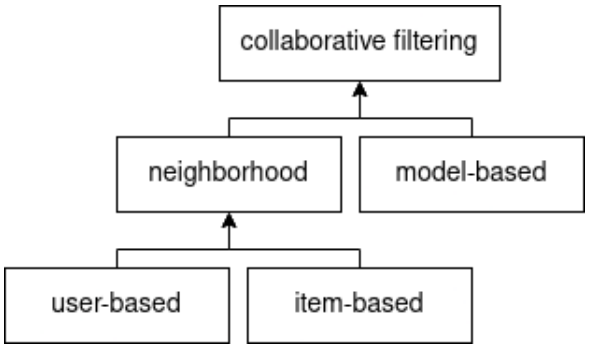

Fig. 3. Unterkategorien von kollaborativen Filtern

gegenüber Modell-basierten System wie z.B. der Einfachheit des Verfahrens, die Tatsache, dass sich die Ergebnisse einfach rechtfertigen lassen, die schnelle Inbetriebnahme und die Tatsache, dass generell stabile Ergebnisse erzielt werden können - auch bei einer stark wachsenden Nutzerzahl, sofern die Ressourcen bereits gestellt werden können. Ein solcher Zuwachs ist vor allem bei kommerziellen Web-Anwendungen bekannt. Aus diesem Grund hat Nachbarschaft-basierte System immer mehr an Bedeutung gewonnen. Nichtsdestotrotz bringt das Verfahren Probleme mit sich. So kann es z.B. vorkommen, dass manche Produkte niemals einem Nutzern empfiehlt wird und dementsprechend das ganze Produktsortiment gar nicht abgedeckt wird. Zudem gibt es das sogenannte Kaltstart-Problem, welches den Zustand beschreibt, dass ein System gar keine bzw. oder nur sehr wenige Bewertungen hat [3].

D. Normalisieren von Bewertungen

Wenn es darum geht, einem Item eine Bewertung zu geben, hat jeder seine eigene persönliche Skala. Selbst wenn eine explizite Definition bei jeder der möglichen Bewertungen angegeben wird, wie beispielsweise 1=”stimme überhaupt nicht zu”, 2=”stimme nicht zu”, 3=”neutral”, usw.), sind sich einige Benutzer unschlüssig. Um diese Schwankung in den persönlichen Bewertungssystem auszugleichen, gibt es verschiedene Verfahren, die bei Empfehlungssysteme eingesetzt werden. Zwei der populärsten Bewertungsnormalisierungsverfahren sind *mean-clustering* und *Z-Score* [3]. Mit diesen wird es Möglich individuelle Bewertungen in eine universellere Skala umzuwandeln. Wie diese beiden Verfahren funktionieren, wird hier jedoch nicht weiter diskutiert, da sie mit dem eigentlichen kNN Verfahren nur bedingt zu tun haben. Dennoch werden wir im späteren Kapitel sehen, wie dieses Konzept zu trage kommen.

II. Data Mining in RE mit kNN

Wie bereits schon in den vorherigen Abschnitten angemerkt, spielt das kNN Verfahren in der Tat eine bedeutsame Rolle in Empfehlungssystemen. Empfehlungssysteme, die auf Nachbarschaft basieren, automatisieren das allgemeine Prinzip, dass ähnliche Benutzer ähnliche Artikel bevorzugen, und ähnliche Artikel von ähnlichen Benutzern bevorzugt werden [3].

Dieses Kapitel ist wie folgt strukturiert. Zunächst wird der Frage auf den Grund gegagen was es genau bedeutet, ein Nachbar innerhalb eines Empfehlungssystem zu sein. Damit geht einher, dass die *Distanz* zwischen zweier Objekte (Benutzer oder Items) berechnet werden müssen. Wie die Berechnung typischerweise in der Praxis geschieht, wird als ersten beschrieben.

In dem darauf folgenden Abschnitt wird erläutert, welche Voraussetzungen benötigt werden, um Regression und Klassifizierung einzusetzen. Denn wie in Kapitel I-A erwähnt, kann kNN für beide Problemstellungen angewendet werden.

Der letzte Teil dieses Kapitels handelt von den zwei Verfahren, welche in Abschnitt I-C angedeutet wurden: *user-based* und *item-based* filtering. Jedes dieser Verfahren wird dabei zunächst beschrieben und anschließend an einem praktischen Beispiel gezeigt. Die Beispiele sowie die Formeln mit deren Variablennamen wurde aus [3] übernommen. Das Beispiel Szenario handelt von Filmen, für welche Bewertungen von verschiedenen Nutzern gegeben wurden. Wie in

der Praxis üblich, kommt es auch in dem Beispiel vor, dass Bewertungen von bestimmten Nutzern fehlen, welche es mittels kNN vorherzusagen gilt. Wird die Bewertung als positiv vorhergesagt, ist das Item ein Kandidat um vorgeschlagen zu werden. Tabelle II zeigt die beispielhaften Bewertungen eines Empfehlungssystems. Als konkreten Nutzer betrachteten wir "Eric" für welchen wir eine Bewertung für den Film Titanic anhand der beiden kollaborativen, Nachbarschaft-basierten Filter Methoden in diesen Abschnitten vorhersagen wollen.

	Matrix	Titanic	Die Hard	F. Gump	Wall-E
John	5	1		2	2
Lucy	1	5	2	5	5
Eric	2	?	3	5	4
Diane	4	3	5	3	

TABLE I
BEISPIEL DATEN

A. Abstandsmessung zum identifizieren von Ähnlichkeiten

In Empfehlungssystemen, sowie bei kNN im Allgemeinen, spielt die Messung der Distanz zweier Punkte zueinander eine fundamentale Rolle. Wie bereits erwähnt, machen Empfehlungssysteme bei Nachbarschaft-basierten Methoden ihre Empfehlung basierend auf dieser Distanz. Die Berechnung der sogenannten Ähnlichkeitsgewichte ist daher einer der der kritischsten Aspekte beim Aufbau eines Nachbarschaft-basierten Empfehlungssystems, da da sie einen signifikanten Einfluss auf die Genauigkeit und die Leistung des Systems haben kann [3]. Für die Messung gibt es verschiedene Ansätze. Einer der wohl am leicht verständlichste ist der *Euklidischer Abstand*:

$$d(x,y) = \sqrt{\sum_{k=1}^{n}(x_k - y_k)^2} \tag{1}$$

Dabei repräsentiert n die Anzahl an Attributen (also der Dimensionen) und x_k und y_k jeweils die k-ten Attribute [2].

Ein weiterer Ansatz der vor allem bei Empfehlungssysteme gängig ist, ist die *Cosine Distance*. Gängige Praxis ist es jedoch, den Abstand direkt in einer Ähnlichkeit auszudrücken. Daher nutzt man gerne die *Sosine Similarity*, welche sich direkt aus der cosine distance ableiten lässt: ($1-$cosine-sim. $=$ cosine-distance). Das liegt der einfachen Annahme zu Grunde, dass wenn die Abstände größer werden, die Ähnlichkeit sinkt und anders rum. Bei der cosine-similarity Methode sind die Input Parameter Vektoren aus einem n-Dimensionen Raum und deren Ähnlichkeit wird anhand des Kosinus des Winkels, den sie bilden:

$$\cos(x,y) = \frac{x \bullet y}{||x||||y||} = \cos(\theta) \tag{2}$$

Dabei repräsentiert $\bullet$ das Skalarprodukt der Vektoren, $||x||$ die Norm des Vektors [2] und θ den Winkel der zwei Punkte. Der Ansatz lässt geometrisch darstellen, wie Bild 4 zeigt. Die dortigen Vektoren befinden sich in einem drei-Dimensionen Raum.

Das Ergebnis ist ein Wert zwischen -1 und 1, wobei 1 bedeutet, dass x y gleich sind. Dementsprechend würde

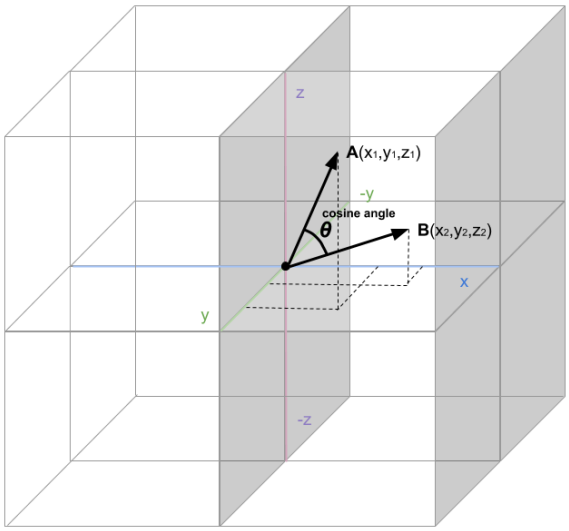

Fig. 4. Geometrische Darstellung der Cosine Similarity [18]

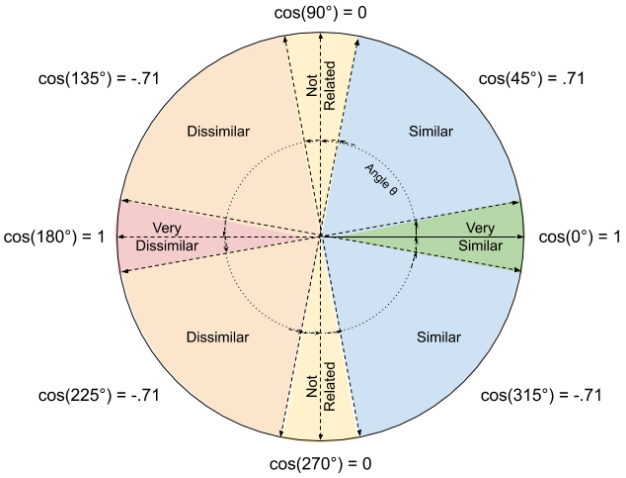

Fig. 5. Interpretation des Ergebnisses der Cosine Similarity [18]

beispielsweise $0,94$ Bedeuten, dass sich die beiden Punkte sehr ähnlich sind. Das folgende Schaubild illustriert die Interpretation von möglichen Ergebnissen der Berechnung.

Ein anderer Ansatz, der ebenfalls gängig bei Empfehlungssystemen ist, ist die Ähnlichkeit anhand der Korrelation anzugeben. Dabei wird die lineare Beziehung zwischen den Items gemessen [2]. Am häufigsten wird hierzu der *Pearson correlation coefficient* verwendet, welche nun kurz eingeführt wird. Gegeben sei die Kovarianz der Datenpunkte x und y und ihre Standardabweichung σ. Man berechnet die Pearson-Korrelation wie folgt.

$$\text{Pearson}(x,y) = \frac{\sum(x,y)}{\sigma_x \times \sigma_y} \tag{3}$$

Unabhängig davon, welche der gerade vorgestellten Methoden benutzt wird um Ähnlichkeiten zu messen, werden die Ergebnisse der Berechnungen typischerweise in einer Matrix festgehalten. Für user-based filtering könnte die Matrix so

aussehen: Bei einen item-based Ansatz würden anstelle der

	John	Lucy	Eric	Diane
John	1,000	-0,938	-0.839	0.659
Lucy	-0.938	1.000	0.922	-0.787
Eric	-0.839	0.922	1.000	-0.659
Diane	0.659	-0.787	-0.659	1,000

TABLE II
BEISPIELHAFTE ÄHNLICHKEITSWERTE [3]

Benutzer die Filme gelistet werden und die Ähnlichkeiten zwischen diesen in die Matrix eingetragen werden.

Wie bereits erwähnt, sieht die Komplexität bei der Berechnung der Ähnlichkeiten eine wichtige Rolle, der kNN ausschließlich im Arbeitsspeicher erfolgt und dementsprechend die Ähnlichkeit unter den Items berechnet werden müssen. Dieses Themenfeld ist immer noch aktuell in der Forschung wie man anhand dieser Arbeit [10] sieht. Dort schlugen Sie eine Methode vor um die Performance von Nachbar-basierten Modellen zu verbessern und erfassten dazu Interaktionen zwischen Benutzern und Objekten, die mit einem traditionellen Ähnlichkeitsmaß nicht erfasst werden können [10]. Auch die Arbeit "Know Your Neighbors (KYN)" [11] beschäftigt sich mit dieser Disziplin. Dort wird ein Verfahren vorgestellt, welches auf einer Kombination etablierter und effizienter maschineller Lernverfahren basiert.

B. Klassifizierung vs. Regression

Wie schon im Abschnitt I-A vorgestellt, kann der kNN Algoithmus sowohl für Regressions- als auch Klassifizierungsprobleme eingesetzt werden. Welche der beiden Methoden am sinnvollsten ist, hängt von der Bewertungsskala ab, welche dem Bewertungssystem zugrunde liegt.

Ist die Bewertungsskala eine kontinuierliche Zahl, wo eine Bewertung beispielsweise einen beliebigen Wert zwischen -10 und 10 annehmen kann, ist Regression die bessere Wahl. Handelt es sich bei der Bewertungsskala allerdings bei einem diskreten Wert wie beispielsweise "gut" oder "schlecht", oder die Werte in der Skala können nicht trivial geordnet werden, dann eine Klassifizierung ist angemessen [3].

C. User-based Recommendation

Ein Empfehlungssystem basierend auf deren Benutzer prognostiziert die Bewertungen r_{ui} (r für eng. "Ratings") eines Benutzers u (für eng. "User") für ein Item i anhand von Bewertungen, welche von Benutzern ähnlich zu u - die "nearest Neighbors", gemacht wurden. Für jeden Benutzer $v \neq u$ nehmen wir an, wir hätten bereits eine Ähnlichkeitswert gegeben, welcher mit w_{uv} gekennzeichnet ist. Die k nächstgelegenen Nachbarn des Nutzers u, also die Benutzer mit den höchsten Ähnlichkeitswert w_{uv}, notieren wir mit $\mathcal{N}(u)$. Da es durchaus sein kann, dass nicht alle Benutzer aus der Menge $\mathcal{N}(u)$ das Item i bewertet haben, filtert man diese zunächst raus. Nur so kann r_{ui} berechnet werden. Diese gefilterte Untermenge notieren wir mit $\mathcal{N}_i(u)$. Nachdem nun alle Komponenten eingeführt wurden, folgt nun ein erster Ansatz zur Berechnung von r_{ui}. Dieser ist die durchschnittliche Bewertung für i von $\mathcal{N}_i(u)$ [3].

$$\hat{r}_{ui} = \frac{1}{|\mathcal{N}_i(u)|} \sum_{v \in N_i} r_{vi} \tag{4}$$

Bei diesem Ansatz sollte schnell ein Problem auffallen. Es wurden zwar nur die k nächstgelegenen Nachbarn des Benutzers u in Betracht gezogen, jedoch wurden die eigentlichen Werte für w_{uv} komplett außer Acht gelassen. Denn obwohl es sich um die nächstgelegenen Nachbarn des Nutzers u handelt, können untereinander gravierende Unterschiede bei der Ähnlichkeit auftreten.

Eine übliche Lösung zu diesem Problem ist es, die Bewertung der Nutzer mit einer Gewichtung zu versehen - den im vorherigen Abschnitt eingeführten Ähnlichkeitswert. Da diese Gewichtungen wahrscheinlich nicht in Summe 1 ergeben, kann es vorkommen dass die vorhergesagte Bewertung außerhalb des Gültigkeitsbereich liegt. In der Praxsis *normalisiert* man die Gewichtungen daher direkt beim Berechnen einer Bewertung, und zwar wie folgt.

$$\hat{r}_{ui} = \frac{\sum\limits_{v \in \mathcal{N}_i(u)} w_{uv} r_{vi}}{\sum\limits_{v \in \mathcal{N}_i(u)} |w_{uv}|} \tag{5}$$

Im Nenner wurde hier der Betrag des Ähnlichkeitswertes benutzt um auszuschließen, dass das Ergebnis außerhalb des Gültigkeitsbereichs liegt. Zu der obigen Formel sei zudem noch gesagt, dass man den Faktor w_{uv} zusätzlich nochmal verstärken kann, indem dieser mit w_{uv}^{α} ausgetauscht wird. Wenn $\alpha > 0$ dient α dabei als Verstärkungsfaktor. Mit $\alpha > 1$ kann man den Nachbarn entsprechend eine noch eine viel größere Relevanz zuzuordnen.

Betrachtet wir nun das zu Beginn des Kapitels eingeführte Beispiel. Angenommen wir wollen die Bewertung für Eric bestimmen und wählen $k = 2$. Lucy und Diane stellen dabei die k-nächstgelegenen Nachbarn dar. In der nachstehenden Tabelle sind die Bewertungen von den zwei Nachbarn markiert, welche für die darauf folgende Berechnung relevant sind.

	Matrix	Titanic	Die Hard	F. Gump	Wall-E
John	5	1		2	2
Lucy	1	5	2	5	5
Eric	2	?	3	5	4
Diane	4	3	5	3	

TABLE III
BEISPIEL EINER USER-BASED RECOMMENDATION

Wir nehmen an, dass Lucy einem Ähnlichkeitswert von 0,75 hat und Diane einen Ähnlichkeitswert von 0,15 hat. Man würde dazu die Werte in die Formel 5 wie folgt einsetzen und kann die Bewertung anschließend ausrechnen.

$$\hat{r}_{\text{Eric, Titantic}} = \frac{0,75 * 5 + 0,15 * 3}{0,75 + 0,15} \simeq 4,67 \tag{6}$$

Man erkennt, dass das Ergebnis näher an Lucy's Bewertung liegt als an Diane's. Nichtsdestotrotz wurde hier noch eine

Sache vergessen - und zwar die Normalisierung der Bewertungen (siehe Kapitel I-D). Das heißt Formel 6 berücksichtigt nicht die Tatsache, dass Benutzer unterschiedliche Bewertungswerte verwenden können, um den gleichen Grad an Wertschätzung für ein Objekt auszudrücken [3]. Das Problem löst man typischerweise mit einer Normalisierungsfunktion $h(r_{vi})$, welche die Bewertungen r_{vi} entsprechend konvertiert. Typischerweise wird dazu eines der beiden aufgelisteten Verfahren in Abschnitt I-D benutzt. Da die Bewertung anschließend wieder zurück konvertiert werden muss, notiert man die Normalisierungsfunktion mit h^{-1}, womit die finale Vorhersage-Formel wie folgt formuliert werden kann:

$$\hat{r}_{ui} = h^{-1}(r_{ui}) \tag{7}$$

Da es bei den gerade eingeführten Berechnungen um kontinuierliche Zahlenwerte ging, haben wir bisher im Grunde ein Regressionsproblem gelöst. Bei einer Nachbarschaft- und Nutzer-basierten Klassifizierung hingegen, wird das Rating eines Benutzers u für ein Item i mittels einer "Abstimmung" ermittelt. Abstimmen können dabei logischerweise nur die k nächstgelegenen Nachbarn. Die Abstimmung, v_{ir} (für eng. vote) für die Bewertung r, welches nun ein konkreter Wert aus einer Menge δ ist, kann wie folgt bestimmt werden:

$$v_{ir} = \sum_{v \in N_i(u)} \delta(r_{vi} = r) w_{uv} \tag{8}$$

Dabei ist $\delta(r_{vi} = r) = 1$, falls $r_{vi} = r$ und ansonsten 0.

Machen wir auch hier ein konkretes Beispiel. Die benötigten Werte dafür sind auch hier die markierten Bewertungen aus Tabelle II-C. Die angenommenen Ähnlichkeitswerte bleiben ebenfalls die selben, also 0,75 für Lucy und 0,15 für Diane. Im folgenden werden wir nur für die Klassen 3,4 und 5 abstimmen lassen. Die Herangehensweise sollte somit schon ausreichend verdeutlicht werden.

$$\begin{aligned} v_{\text{titanic},3} &= \delta(5=3)*0,75 + \delta(3=3)*0,15 \\ &= 0*0,75 + 1*0,15 = 0,15 \end{aligned}$$

$$\begin{aligned} v_{\text{titanic},4} &= \delta(5=4)*0,75 + \delta(3=4)*0,15 \\ &= 0*0,75 + 0*0,15 = 0 \end{aligned}$$

$$\begin{aligned} v_{\text{titanic},5} &= \delta(5=5)*0,75 + \delta(3=5)*0,15 \\ &= 1*0,75 + 0*0,15 = 0,75 \end{aligned}$$

In dem Beispiel gab ein nur zwei Stimmen. Eine Stimme für 5 von Lucy und eine Stimme für 3 von Diane. Durch die Tatsache, dass Lucys einen höhren Ähnlichkeitswert zu Eric hat als Diane, gewinnt Lucys Vote, womit das Ergbenis 5 ist. Das Ergbenis ist hier also das selbe, wie bei der Regression - das muss aber in der Regel nicht der Fall sein.

Es sei bemerkt, dass auch bei der Klassifizierung die Bewertungen normalisiert werden müssten, genau wie bei der Regression.

$$\hat{r}_{ui} = h^{-1}(\arg\max_{r \in \mathcal{S}'} \sum_{v \in \mathcal{N}_i(u)} \delta(h(r_{vi} = r) w_{uv})) \tag{9}$$

D. Item-based Recommendaion

Der nächste Abschnitt erläutert die zweite Nachbarschaftsbasierte Methode um Bewertungen vorherzusagen - die Item-basierte Empfehlung. Anstatt sich auf ähnliche Benutzer zu konzentrieren, legt man bei dieser Methode einen Fokus auf die Ähnlichkeit der eigentlichen Items. Die Nachbarn des kNN Verfahren sind hier also die Items untereinander und nicht die Benutzer. In unserem Beispiel betrachten wir nun die Filme, die Erik bereits gesehen hat und wozu er eine Bewertung abgegeben hat und darauf aufbauen eine Bewertung für Titanic vorherzusagen. Das Empfehlungssystem würde dabei erkennen, dass Benutzer welche den Film Titanic bereits bewertet haben, ähnliche Bewertungen den Filmen "Forrest Gump" und "Wall-E" gegeben haben. Aufgrund der Tatsache dass Erik diese Filme ebenfalls mag, würde das Empfehlungssystem annehmen, dass dieser auch Eric gefallen würde.

Die Vorgehensweise wird nun formal erläutert. Seien $\mathcal{N}_u(i)$ die Nachbarn eines Items i, für welche ein Nutzer u Bewertungen abgegeben hat. Mit der folgenden Formel lässt sich eine Bewertung eines Items mittels Regression für einen Nutzer vorhersagen.

$$\hat{r}_{ui} = \frac{\sum_{j \in N_u(i)} w_{ij} r_{uj}}{\sum_{j \in N_u(i)} |w_{ij}|} \tag{10}$$

Das lässt sich wieder anhand eines Beispiels verdeutlichen. Grundlage dafür sollen nochmals die Daten aus Tabelle II sein. Wieder wählen wir k=2 und nehmen an, die nächstgelegenen Nachbarn seien Forrest Gump mit einem Ähnlichkeitswert von 0,85, sowieso Wall-E mit einem Ähnlichkeitswert von 0,75. Die entsprechenden Bewertungen zur Berechnung findet man dementsprechend in den gekennzeichneten Feldern in Tabelle IV. Somit hat man alle Werte für die Formel 10 und kann

	Matrix	Titanic	Die Hard	F. Gump	Wall-E
John	5	1		2	2
Lucy	1	5	2	5	5
Eric	2	?	3	5	4
Diane	4	3	5	3	

TABLE IV
TABLE TO TEST CAPTIONS AND LABELS

diese einsetzen:

$$\hat{r} = \frac{0,85*5 + 0,75*4}{0,85 + 0,75} \simeq 4,53 \tag{11}$$

Wie bereits angedeutet, muss die Bewertungsskala noch mittels einer Funktion h angepasst werden.

Auch bei der Item-basierten Herransgehensweise gibt kann man anstelle von Regression auch Bewertungen klassifizieren. Die Formel dafür ist wie folgt.

$$v_{ir} = \sum_{j \in N_u(i)} \delta(r_{uj} = r) w_{ij} \tag{12}$$

Anhand der Beispieldaten berechnen wir auch hier die Item-basierten Bewertungen von Eric für den Film Titanic.

$$\begin{aligned} v_{\text{titanic},3} &= \delta(5=3)*0,85+\delta(4=3)*0,75 \\ &= 0*0,85+0*0,75=0 \end{aligned}$$

$$\begin{aligned} v_{\text{titanic},4} &= \delta(5=4)*0,85+\delta(4=4)*0,75 \\ &= 0*0,85+1*0,75=0,75 \end{aligned}$$

$$\begin{aligned} v_{\text{titanic},5} &= \delta(5=5)*0,85+\delta(3=5)*0,75 \\ &= 1*0,85+0*0,75=0,85 \end{aligned}$$

Das Ergebnis der Abstimmung ist hier wieder, dass die Bewertung der Klasse "5" zugeordnet wird. Nachstehend ist die finale Formel zur Berechnung der Klasse unter Berücksichtigung einer Normalisierung aufgeführt.

$$\hat{r}_{ui} = h^{-1}(\arg\max_{r\in\mathcal{S}'} \sum_{j\in\mathcal{N}_u(i)} \delta(h(r_{uj}=r)w_{ij})) \tag{13}$$

E. Nutzer-basiert oder Item-basiert

Welches der beiden grade vorgestellten Verfahren man wählen sollte, hängt von mehreren Faktoren ab. Laut [3] sollten fünft Kriterien berücksichtigt werden.

1) Genauigkeit: Die Genauigkeit von Nachbarschafts-basierten Empfehlungssystemen basiert zu einem Großteil auf dem Verhältnis zwischen der Anzahl von Benutzern und der Anzahl an Bewertungen [3]. Generell ist es stets vorzuziehen, eine kleine Anzahl an Nachbarn zu identifizieren, bei denen man mit Garantie weiß, dass diese untereinander ähnlich sind, als einer großen Menge an Nachbarn, bei welchen kaum Sicherheit besteht, dass diese viele Ähnlichkeiten untereinander haben. In Fällen, bei denen die Anzahl der Benutzer wesentlich größer ist als die Anzahl der Items (wie beispielsweise bei großen kommerziellen Systemen wie Amazon.com), können Item basierende Methoden daher genauere Empfehlungen konstruieren. Gleiches gilt für Systeme, die weniger Benutzer als Items haben, z.B. ein Empfehlungssystem für Forschungsarbeiten mit mit Tausenden von Benutzern, aber Hunderttausenden von zu empfehlenden Artikeln, können mehr von Benutzer-basierten Nachbarschaftsmethoden profitieren [3].

2) Effizienz: Nicht nur die Genauigkeit der Vorhersagen, sondern auch der Resourcenbedarf hängt von dem Verhältnis zwischen der Anzahl an Benutzern und der Anzahl an Items zusammen. Ist die Anzahl an Benutzern größer als die Anzahl an Items (wie es oft der Fall ist in der Praxis), Item-basierte Empfehlungssysteme benötigen weniger Speicher und und Rechenaufwand für das Berechnen Ähnlichkeitswerte, also schlussendlich dem Berechnen der Nachbarn [3]. Ist die Anzahl an Benutzern kleiner als die der Items, ist ein Benutzer-basiertes Empfehlungssystem Resourcensparender. Nichtsdestotrotz gelten Item-basierte Empfehlungssysteme als besser skalierbar, da die Anzahl an Benutzern ohne Probleme wachsen kann und man bei Schwankungen hinsichtlich Benutzerzahlen keine unnötigen Ressourcen verbrauchen muss.

3) Stabilität: Die grade angedeuteten Schwankungen spielen ebenfalls eine Rolle bei der Wahl zwischen Benutzer-basierten Empfehlungssystemen und Item-basierten Empfehlungssystemen. Sind die verfügbaren Items eher statisch im Vergleich zu den Benutzern des Systems, sollte eine Item-basierte Methode bevorzugt werden. So können die Ähnlichkeitswerte der Items in unregelmäßigen Abständen berechnet werden können und das System dennoch in der Lage ist Items an neue Benutzer zu empfehlen. Verändern sich die Items jedoch regelmäßig, beispielsweise bei einem Online Nachrichtenportal, Benutzer-basierte Empfehlungssysteme wären wahrscheinlich stabiler [3].

4) Rechtfertigung und Anpassbarkeit: Ein Alleinstellungsmerkmal von Item-basierten Methoden ist, dass die Ergebise von diesen leicht zu rechtfertigen sind. So kann die Liste der Nachbarn eines Items, sowie deren Ähnlichkeitswerte dem Benutzer als Erklärung der Empfehlung präsentiert werden. Durch Ändern der Liste der Nachbarn und/oder der Ähnlichkeitensgewichtung kann der Benutzer interaktiv am Empfehlungsprozess teilnehmen. Benutzerbasierte Verfahren sind für diesen Prozess jedoch weniger geeignet, weil der aktive Benutzer die anderen Benutzer, die als Nachbarn in der Empfehlung dienen, nicht kennt [3].

5) Neue Entdeckungen: In einem Item-basierten Empfehlungssystemen werden die Bewertungen für ein Item anhand von Bewertungen zu ähnlichen Artikeln gemacht. Daraus folgt, dass diese Systeme dazu neigen, Items zu empfehlen, welche der Benutzer bekanntermaßen mag. Zum Beispiel in einer Filmempfehlungsanwendung werden Filme, die das Genre, die Schauspieler oder den Regisseur haben, die ein Benutzer mag, empfohlen. Dies sind zwar "sichere" Empfehlungen, aber es hilft dem Benutzer weniger, neue Schauspieler oder Regisseure zu entdecken, die ihm gefallen könnten. Was diese Disziplin angeht, so würden Bernutzer-basierte Empfehlungssysteme eher für Neuentdekcungen sorgen [3].

III. Stand der Forschung

kNN ist aus den Empfehlungssystemen schon jetzt nicht mehr wegzudenken. Durch seine Einfachheit wird er zunehmen in der Industrie eingesetzt. Nichtsdestotrotz bietet er noch weiteres Forschungspotential welches Wissenschaftler nutzen. Die Zahl an neuen Publikationen in dem Gebiet ist enorm. Dieses Kapitel soll einen Einblick in den aktuellen Forschungsstand bieten indem einige Arbeiten hier kurz vorgestellt werden.

Ein Problem mit dem kNN Algorightmus ist die Skalierung. Wie bereits in I-A erwähnt, ist der Algorithmus ein sog. *Layz Learner*, da er seine Berechnungen bezüglich der Entfernungen zu den Objekten untereinander stets im Arbeitsspeicher halten muss. Traditionelle Verfahren, wie sie in dieser Arbeit vorgestellt wurden, können dementsprechend bei großen Datenmengen zu unpraktikablen Rechenkosten führen. Diesem Problem geht z.B. diese Arbeit [12]

nach und stellen dazu eine Erweiterungen vor des kNN Verfahrens vor. In dieser Erweiterung werden Benutzer- und Item-Nachbarschaften in einem ersten Offline-Schritt vorberechnet und mit Hilfe einer entsprechenden Datenstruktur gespeichert. Die gespeicherten Nachbarschaftsinformationen ermöglichen es dann spätere Berechnungen zu Empfehlungen zu beschleunigen [12]. Diese Variante testeten die Forscher an einem großen Beispieldatansatz und konnten somit eine erhöhte Effizienz von kNN im Vergleich zu den State-of-the-Art Methoden beweisen.

Auch in dieser Publikation [13] wurde eine Empfehlungsstrategie gezeigt, welche als eine Variante des Nearest-Neighbors-Schemas gesehen werden kann. Dazu statteten sie das Verfahren mit einer "stochastischen Erkundungsfunktion" aus, welche die Nachbarschaft eines Benutzers ausfindig macht. Dabei nutzen die Authoren das sog. *Thompson Sampling*.

Ebenso wie Erweiterungen des kNN gibt es auch Ansätze, kNN mit anderen Maschine Learning Techniken zu verbinden. Solch eine hybride Methode wird beispielweise in dieser Arbeit [14] vorgestellt. Dabei beschreibten die Forscher einen neuen Ansatz um einen nächsten Musiktitel bei Musik Streaming Plattformen zu empfehlen. Die benutzten dazu eine Kombination aus Nearest-Neighbor-Techniken, einem Standard-Matrixfaktorisierungs-Algorithmus und Heuristiken.

Ebenso wie neue Erweiterungen publiziert werden, werden auch immer mehr Bereiche mit einem Empfehlungssystem druch Nachbarschafts-basierte Ansätze abgedeckt. So sind es mittlerweile nicht nur kommerzielle Systeme die Empfehlungen aus wirtschaftlichen Gründen benutzen, sondern werden Empfehlungssystem für eine Vielzahl an Anwendungsgebieten interessant. In dieser Publikation [15] wurde beispielsweise eine auf kNN basiertes Empfehlungssystem vorgestellt, mit welchem man Forschungsgruppen aufstellen könnte, indem Mitwirkende zu einem bestimmten Forschungsgebiet empfohlen werden. Eine andere Arbeit [16] veröffentliche einen Vorschlag zur Überwachung von Vulkanen, welches Empfehlungen hinsichtlich der Sperrung von Gebieten machen kann. Dabei spielte ebenfalls ein nachbarschaftsbasierte Methode eine Rolle.

Bei Empfehlungssystemen im Allgeminen gibt es noch einige Probleme zu lösen. Eine Auflistung dieser Probleme fasst diese Arbeit [9] unter anderem zusammen. Zur Lösung einiger dieser Probleme könnten Varianten des kNN Algorithmus helfen und Publikationen in dieser Richtung sind zu erwarten.

IV. Fazit

Die Literaturrechere hat ergeben, dass der kNN Algorithmus einen enormen Stellenwert bei Empfehlungssystemen hat. Sogenannte Nachbarschafts-basierte Verfahren sind gängige Praxis bei in Empfehlungssystemen und lässt sich dementsprechend hervorragend mit dem kNN Algorithmus verbinden. Wie das in praktisch gemacht wird, wurde in dieser Arbeit formal beschrieben und anhand eines Beispiels vorgeführt.

Aufgrund der Tatsache, dass kNN keine Trainingsphase hat, also ein sog. *Layz Learner* ist, müssen zur Ermittelung der Nachbarn eines neuen Elements, alle Distanzen zu den anderen Elementen aus dem Datensatz berechnet werden. Je Anazhl der Benutzer oder Objekte in einem Empfehlungssystem, kann das erhebliche Resource in Anspruch nehmen. Zusätzlich sind die Distanz-Berechnungen ausschlaggebend für die Komplexität von kNN und kann bei großen Datenmengen ebenfalls zu Performance-Problemen führen. Durch die Wahl der richtigen Methode (user-based oder item-based) kann man diese Probleme adressieren. Die Forschung ist bezüglich aktive und versucht die Probleme anhand von Erweiterungen des kNN und alternative Methoden zur Berechnung der Distanzen zu adressieren.

References

[1] Ricci F., Rokach L., Shapira B. (2015) Recommender Systems: Introduction and Challenges. In: Ricci F., Rokach L., Shapira B. (eds) Recommender Systems Handbook. Springer, Boston, MA. https://doi.org/10.1007/978-1-4899-7637-6_1

[2] Amatriain X., Jaimes* A., Oliver N., Pujol J.M. (2011) Data Mining Methods for Recommender Systems. In: Ricci F., Rokach L., Shapira B., Kantor P. (eds) Recommender Systems Handbook. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-85820-3_2

[3] Ning X., Desrosiers C., Karypis G. (2015) A Comprehensive Survey of Neighborhood-Based Recommendation Methods. In: Ricci F., Rokach L., Shapira B. (eds) Recommender Systems Handbook. Springer, Boston, MA. https://doi.org/10.1007/978-1-4899-7637-6_2

[4] Koren Y., Bell R. (2015) Advances in Collaborative Filtering. In: Ricci F., Rokach L., Shapira B. (eds) Recommender Systems Handbook. Springer, Boston, MA. https://doi.org/10.1007/978-1-4899-7637-6_3

[5] R. Burke, "Hybrid Web Recommender Systems", 2007, The Adaptive Web: Methods and Strategies of Web Personalization, pp. 377–408, Springer Berlin Heidelberg, doi: 10.1007/978-3-540-72079-9_12

[6] Jannach D., Zanker M., Ge M., Gröning M. (2012) "Recommender Systems in Computer Science and Information Systems – A Landscape of Research", Huemer C., Lops P. (eds) E-Commerce and Web Technologies. EC-Web 2012. Lecture Notes in Business Information Processing, vol 123. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-32273-0_7.

[7] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," in IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 6, pp. 734-749, June 2005, doi: 10.1109/TKDE.2005.99.

[8] R. Burke, "Hybrid Recommender Systems: Survey and Experiments", 2002, User Modeling and User-Adapted Interaction, pp. 331-370, doi: 10.1023/A:1021240730564

[9] F. Ricci, "Recommender Systems: Models and Techniques", 2017 Springer New York, Encyclopedia of Social Network Analysis and Mining, doi: 10.1007/978-1-4614-7163-9_88-1

[10] V. W. Anelli, T. Di Noia, E. Di Sciascio, A. Ragone, and J. Trotta. 2019. "The importance of being dissimilar in recommendation". In Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing (SAC '19). Association for Computing Machinery, New York, NY, USA, 816–821. DOI:https://doi.org/10.1145/3297280.3297360.

[11] J. Misztal-Radecka and B. Indurkhya, 2020, "Getting to Know Your Neighbors (KYN). Explaining Item Similarity in Nearest Neighbors Collaborative Filtering Recommendations." In Adjunct Publication of the 28th ACM Conference on User Modeling, Adaptation and Personalization (UMAP '20 Adjunct). Association for Computing Machinery, New York, NY, USA, 59–64. DOI:https://doi.org/10.1145/3386392.3397599.

[12] A. Sagdic, C. Tekinbas, E. Arslan and T. Kucukyilmaz, "A Scalable K-Nearest Neighbor Algorithm for Recommendation System Problems," 2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO), 2020, pp. 186-191, doi: 10.23919/MIPRO48935.2020.9245195.

[13] J. Sanz-Cruzado, P. Castells, and E. López. 2019. "A simple multi-armed nearest-neighbor bandit for interactive recommendation". In Proceedings of the 13th ACM Conference on Recommender Systems (RecSys '19). Association for Computing Machinery, New York, NY, USA, 358–362. DOI:https://doi.org/10.1145/3298689.3347040.

[14] M. Ludewig, I. Kamehkhosh, N. Landia, and D. Jannach. 2018. "Effective Nearest-Neighbor Music Recommendations". In Proceedings of the ACM Recommender Systems Challenge 2018 (RecSys Challenge '18). Association for Computing Machinery, New York, NY, USA, Article 3, 1–6. DOI:https://doi.org/10.1145/3267471.3267474.

[15] C. Yang, T. Liu, L. Liu and X. Chen, "A Nearest Neighbor Based Personal Rank Algorithm for Collaborator Recommendation," 2018 15th International Conference on Service Systems and Service Management (ICSSSM), 2018, pp. 1-5, doi: 10.1109/ICSSSM.2018.8465112.

[16] F. Tempola, A. Arief and M. Muhammad, "Combination of case-based reasoning and nearest neighbour for recommendation of volcano status," 2017 2nd International conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE), 2017, pp. 348-352, doi: 10.1109/ICITISEE.2017.8285525.

[17] B. Li, S. Wan, H. Xia and F. Qian, "The Research for Recommendation System Based on Improved KNN Algorithm," 2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications(AEECA), 2020, pp. 796-798, doi: 10.1109/AEECA49918.2020.9213566.

[18] Don Cowan, Online-Resource, 2021, https://www.ml-science.com/cosine-similarity