

Fachhochschule Aachen, Campus Jülich
Fachbereich 09: Medizintechnik und Technomathematik
Studiengang Scientific Programming, B. Sc.

**Vergleich unterschiedlicher
Bewertungsmetriken in sozialen Medien**
**Eine agent*innenbasierte Modellierung von
Votingsystemen in der Onlinekommunikation**

Bachelorarbeit

von

Felix Matuschka

Matr.-Nr.: 3169962

betreut von

1. Prüfer: Prof. Dr. Alexander Voß
2. Prüfer: Dr. André Calero-Valdez

Eidesstattliche Versicherung

Diese Arbeit ist von mir selbständig angefertigt und verfasst. Es sind keine anderen als die angegebenen Quellen und Hilfsmittel benutzt worden.

Aachen, 16. August 2020

Felix Matuschka

Inhaltsverzeichnis

Formelzeichen und Symbole	i
1. Einleitung	2
1.1. Motivation	2
1.2. Agent*innenbasierte Modellierung	3
2. Verwandte Arbeiten	4
2.1. agent*innenbasierte Modellierung von Meinungsdynamiken	4
2.2. Bewertungsmetriken und Fairness	4
2.2.1. Fairness mit Gruppen	5
2.2.2. Individuelle Fairness in sozialen Medien	5
3. Modell von Votingsystemen	7
3.1. Definition der Qualität	7
3.2. Bewertungsraum des Votingsystems	8
3.3. Posts	8
3.3.1. Beobachtbare Parameter	8
3.3.2. Nicht beobachtbare Parameter	9
3.4. Bewertungsmetriken	9
3.5. User*innen	9
3.6. User*innen- und Postanzahl	12
4. User*innenmeinung	13
4.1. Transformation der Qualitätsparameter	13
4.2. Approximation der User*innenmeinungsfunktion	13
4.2.1. Meinung im Konsens	14
4.2.2. Meinung im Dissens	14
5. Bewertungsmetriken	16
5.1. Bewertungstransformation	16
5.1.1. Differenz	16
5.1.2. Anteil	17
5.1.3. Wilson Score	17
5.2. Bewertungsmetriken	17
5.2.1. Hacker News Metrik	17
5.2.2. Verallgemeinerte Hacker News Metrik	17
5.2.3. Reddit Hot Metrik	18
5.2.4. Viewmetrik	18
5.2.5. Aktivitätsmetrik	19
5.3. Zufallsbewertung	19
5.3.1. μ -Abweichung	19
5.3.2. σ -Abweichung	19

6. Evaluationsmethode	20
6.1. Iterationsevaluation	20
6.1.1. Discounted Cumulative Gain	20
6.1.2. Gini-Koeffizient	21
6.2. Modellevaluation	21
6.2.1. Aggregation von Iterationsevaluationsfunktionen	21
6.2.2. Posts ohne Betrachtungen	21
7. Implementierung	23
7.1. agent*innenbasierte Modellierung von Votingsystemen	23
7.1.1. Agent*innenschrittfunktion	24
7.1.2. Modellschrittfunktion	24
7.2. Erstellung von Modellkonfigurationen	25
7.3. Auswertung der Modelle	26
8. Simulationsergebnisse	28
8.1. Größe des Bewertungsvektor	29
8.2. Initialscore	29
8.3. Gravität	30
8.4. Bewertungstransformation	30
8.5. Zufallsabweichung	32
8.6. Modell- und User*innenparameter	32
8.6.1. Iterationslänge	32
8.6.2. Qualitätsraum	34
8.6.3. User*innenparameter	34
8.6.4. User*innen- und Postanzahl	35
9. Diskussion der Ergebnisse	36
10. Fazit	38
11. Ausblick	38
Literatur	40
A. Modellkonfigurationen	42

Zusammenfassung Die riesige Anzahl an Beiträgen auf sozialen Medien ist für Menschen nicht überschaubar. Um Nutzer*innen die interessanten und relevanten Beiträge anzuzeigen müssen diese vorsortiert werden. Für die Sortierung werden Bewertungsmetriken verwendet, diese sortieren Posts anhand ihrer Eigenschaften, zum Beispiel der Anzahl und Art der Bewertungen, die sie von den Nutzer*innen erhalten haben.

In dieser Arbeit wird ein agent*innenbasiertes Modell von Votingsystemen in der Onlinekommunikation entwickelt, mithilfe dessen unterschiedliche Bewertungsmetriken nach einem definierten Fairnessbegriff verglichen werden. Dazu werden einige Bewertungsmetriken beschrieben und auf unterschiedlichen Arten von Plattformen, wie News- und Q&A-Plattformen, in der Simulation einer Onlinekommunikationsplattform angewendet und ausgewertet. Die getesteten Bewertungsmetriken unterscheiden sich in ihrer Fairness. Die Wahl der Bewertungsmetrik hat somit einen relevanten Einfluss auf das soziale Medium.

Formelzeichen und Symbole

Modellparameter

M_N	Anzahl der Iterationen des Modells
i_M	Iterationsschritt des Modells
P	Post
U	User*in
O	User*innenmeinungsfunktion
W	User*innenmeinungsverteilung
V	Bewertungsraum eines Votingsystems
R_γ	Gravität der Relevanz
U_N	Anzahl der User*innen
Q	Qualitätsverteilung
P_{start}	Startpost
P_{iter}	neuer Post in Iteration

Postparameter

q_P	Qualität
r_P	Relevanz
t_P	Veröffentlichungszeitpunkt
b_P	Bewertungsvektor
v_P	Anzahl der Betrachtungen
s_P	Score
$n_{\uparrow P}$	Anzahl der Upvotes
$n_{\downarrow P}$	Anzahl der Downvotes

User*innenparameter

qu	Qualitätsperzeption
k_U	Konzentration
z_U	Bewertungszufriedenheit
a_U	Aktivität
$m_{P,U}$	Meinungswert bezüglich des Posts P

Parameter der Bewertungsmetriken

B	Bewertungsmetrik
τ	Transformation des Bewertungsvektors
γ	Gravität
t_0	Initialscore von Posts

Evaluationsfunktionen

G	Gini-Koeffzient
DCG	Discounted Cumulative Gain
$nDCG$	Normalized Discounted Cumulative Gain
$P_{v=0}$	Anzahl von Posts ohne Betrachtungen
T	Trapezregel

Index

N	Länge eines Vektors oder Dimension
-----	------------------------------------

1. Einleitung

1.1. Motivation

Im letzten Jahrzehnt sind soziale Medien zu einem wichtigen Teil in der Kommunikation herangewachsen. Im Gegensatz zu konventionellen Medien können Nutzer*innen von sozialen Medien selbst Inhalte veröffentlichen, dadurch gibt es es viel mehr Beitragende als je zuvor. Dies führt zu einer Menge an Information, die kein Mensch überblicken oder verstehen kann. Von Social Media Plattformen werden Algorithmen verwendet, welche die Beiträge filtern und den Nutzer*innen diejenigen anzeigen, die sie wahrscheinlich interessant finden [Sar+01].

Um Informationen darüber zu generieren, welche Beiträge für Nutzer*innen relevant sind, verwenden viele soziale Medien und Onlineplattformen *Votingsysteme*, um Nutzer*innen die Möglichkeit zu geben ihre Meinung über einzelne Beiträge zu äußern. Hacker News¹ erlaubt Nutzer*innen Posts nur positiv, mit Upvotes, zu bewerten. Auf Reddit² ist es ebenfalls möglich Posts mit Downvotes herabzuwerten. Onlineshops wie TheCubicle³ verwenden zum Beispiel ein 1 bis 5 Sterne System um Artikel bewerten zu lassen.

Um Nutzer*innen Beiträge personalisiert anzuzeigen können Empfehlungssysteme verwendet werden. Diese zeigen jene Beiträge an, welche andere Nutzer*innen mit ähnlichen Interessen bereits als gut bewertet haben. Während Nutzer*innen aus ihrer Perspektive interessante Posts erhalten, hat diese Methode einige negative Effekte. Die Anwendung, so die Theorie, führt zur Bildung von Filterblasen [FGR16]. Nutzer*innen interagieren nur noch mit Beiträgen, die sie positiv wahrnehmen. Nutzer*innen, die sich zum Beispiel in der Filterblase eines politischen Extrems befinden, erhalten so keine kritischen Beiträge mehr zu ihrer eigenen Meinung, wodurch sich diese verstärkt.

Filterblasen können vermieden werden, indem allen Nutzer*innen die gleichen Beiträge angezeigt werden. Dennoch müssen die Beiträge anhand des Interesses für Nutzer*innen gefiltert und sortiert werden. Dazu werden *Bewertungsmaetriken* verwendet, die die Posts anhand ihrer Eigenschaften, zum Beispiel unter Einbeziehung ihrer Upvotes und des Veröffentlichungszeitpunkt, bewerten und anordnen.

Eine Problematik, die allerdings auch bei Bewertungsmaetriken auftritt, ist der *Matthäus-Effekt*: Beiträge, die bereits viel Aufmerksamkeit erhalten haben, werden deshalb weiteren Nutzer*innen angezeigt und erhalten dadurch noch mehr Aufmerksamkeit und Interaktion. Beiträge hingegen, die wenig Aufmerksamkeit erhalten, werden durch die Bewertungsmaetrik

¹<https://news.ycombinator.com>

²<https://www.reddit.com>

³<https://www.thecubicle.com>

1. Einleitung

oder das Empfehlungssystem nicht weiter als interessant erachtet und keinen weiteren Nutzer*innen angezeigt [Sto15].

Im Gegensatz zu Empfehlungssystemen, basieren Bewertungsmaßnahmen meist nicht auf Methoden des maschinellen Lernens, sondern auf einfachen Algorithmen, die leicht austauschbar sind [AT05].

Es ist wünschenswert eine Bewertungsmaßnahme zu finden, die die Matthäus-Effekte minimiert. Aufgrund ihrer einfachen Struktur können sie leicht simuliert und verglichen werden. Es ist nicht ausgeschlossen, dass das Votingssystem Einfluss auf die Matthäus-Effekte hat. Das Votingsystem sollte auf der Suche nach einer optimalen Bewertungsmaßnahme mit einbezogen werden.

In dieser Arbeit wird ein agent*innenbasiertes Modell von Votingsystemen in sozialen Medien entworfen, um unterschiedliche Bewertungsmaßnahmen zu vergleichen.

1.2. Agent*innenbasierte Modellierung

In der agent*innenbasierten Modellierung werden die Aktionen und Interaktionen einzelner Entitäten, der Agent*innen, simuliert. Auch die Umwelt, in der die Agent*innen interagieren, wird modelliert. Mit Agent*innen können unterschiedliche Dinge beschrieben werden. Sie sind nicht auf die Modellierung von Menschen beschränkt. Durch Agent*innen können zum Beispiel in einer Simulation eines Vogelschwarmes die Vögel modelliert werden. Für die Modellierung eines Waldbrandes würden die Agent*innen Bäume repräsentieren. Einzelne Agent*innen können dabei in ihren Eigenschaften sehr unterschiedlich sein. Werden Menschen durch die Agent*innen modelliert, können diese zum Beispiel extrovertiert oder introvertiert sein, eigennützig oder im Sinn der Gruppe handeln.

Wie in [Bur+20] beschrieben sind agent*innenbasierte Modelle weder absolut realistisch noch vollständig. Durch sie wird eine vereinfachte Realität modelliert. Die Agent*innen können zwar beliebig komplex parametrisiert werden, dennoch werden meist sehr einfache Verhaltensregeln der Agent*innen definiert. Denn durch die Interaktion entstehen bereits hier komplexe Systeme, welche die Realität ausreichend approximieren.

Kleine Veränderungen am Verhalten der Agent*innen können bereits große Veränderungen des Simulationsergebnisses hervorrufen. In der Simulation treffen die Agent*innen individuell unterschiedliche Entscheidungen, die durch die Verhaltensregeln vorgegeben sind und durch die Wahrnehmung der Agent*innen der Umwelt beeinflusst wird.

Die Agent*innen können in einem Netzwerk angeordnet sein, sodass sie sich gegenseitig in ihrem Verhalten beeinflussen, sie können jedoch auch nur mit dem System, in dem sie sich befinden, interagieren.

Die Popularität von agent*innenbasierten Modellen steigt stetig, sie sind beliebt, um unter anderem biologische, ökonomische und soziale Systeme, wie zum Beispiel soziale Medien zu modellieren. Auch in dieser Arbeit wird die agent*innenbasierten Modellierung zur Simulation einer Social Media Plattform verwendet.

2. Verwandte Arbeiten

In dieser Arbeit wird ein agent*innenbasiertes Modell eines sozialen Mediums erstellt. Dafür ist es interessant, die bisherigen verwandten Anwendungen von agent*innenbasierte Modellierung zu überblicken. Es zeigt sich, dass diese Art der Modellierung viel Anwendung in der Simulation von Meinungsbildungsprozessen und Meinungsdynamiken findet.

Viele Arbeiten beschäftigen sich mit der Fairness von Sortierungen in sozialen Medien. Einige Erkenntnisse und Ideen der Arbeiten können zur Untersuchung der Bewertungsmetriken in dieser Arbeit verwendet werden und in diesem Kapitel vorgestellt.

2.1. agent*innenbasierte Modellierung von Meinungsdynamiken

Um Meingungsdynamiken und Meinungsbildungsprozesse zu modellieren werden häufig agent*innenbasierte Modelle verwendet. Dabei werden Agent*innen als meinungshabende Individuen modelliert, die andere Agent*innen in ihrer Meinung beeinflussen und von anderen selbst beeinflusst werden. [MVN19] ist eine bibliographische Übersicht, in der die wichtigsten Charakteristiken solcher Modelle herausarbeitet werden.

Für eine Gruppe aus Individuen mit eigener subjektiver Wahrnehmung wird von [Deg74] ein Modell vorgeschlagen, das die unterschiedlichen Meinungen zu einem Konsens zusammenführen kann.

In [TL14] wird ein agentenbasiertes Modell mit skeptizistischen Agent*innen gegenüber anderen Meinungen vorgestellt. Es wird gezeigt, dass auch in solchen Konstellationen ein Konsens gefunden werden kann.

2.2. Bewertungsmetriken und Fairness

Einige Arbeiten beschäftigen sich mit Fairness zwischen *beschützten* Gruppen, den Minderheiten, und *nicht geschützten* Gruppen, den Mehrheiten. Es wird versucht eine Benachteiligung der *geschützten* Gruppen zu vermeiden. Anhand von Experimenten wird erforscht wie sich sozialer Einfluss auf individuelle Fairness in Bewertungsmetriken auswirkt. Für bekannte soziale Medien werden die Bewertungsmetriken analysiert.

2.2.1. Fairness mit Gruppen

In [Cas19] stellen die Autor*innen fest, dass Fairness zwischen Gruppen hergestellt ist, wenn der Quotient aus Sichtbarkeit und Qualität von Elementen der Gruppen gleich ist.

[SJ18] und [YS17] betrachten Fairness in Sortierungen unter Einbeziehung von *beschützten* und *nicht beschützten* Objekten. Es werden Messfunktionen für Fairness eingeführt, um die Benachteiligung von *beschützten* Objekten systematisch auszuwerten.

Ein Algorithmus, der aus einer Menge von *beschützten* und *nicht beschützten* Objekten die qualitativsten k Objekte sucht wird, in [Zeh+17] beschrieben. Für diese k Objekte sind *nicht beschützte* Objekte nicht benachteiligt.

Gruppenbezogene Fairness wird in [BGW18] nur als Spezialfall von individueller Fairness betrachtet. Es wird der *Discounted Cumulative Gain*-Koeffizient (*DCG*) zur Messung der Fairness der Bewertungsmetriken verwendet. Die Autor*innen stellen fest, dass sich der *DCG* mit einer einzelnen Bewertungsmetrik nicht optimieren lässt. Sie stellen ein Optimierungsproblem auf, welches durch die mehrfache Anwendung von unterschiedlichen Bewertungsmetriken Fairness herstellt.

2.2.2. Individuelle Fairness in sozialen Medien

Die Autoren von [CSV18] argumentieren, dass viele Bewertungsmetriken die Diversität verringern und Stereotypen reproduzieren, die Bewertungsmetriken jedoch nicht darauf geprüft werden und die Stereotypen sich dadurch verstärken. Es wird ein Algorithmus vorgeschlagen, welcher Elemente nach bestimmten Fairnessbedingungen anordnet.

In [SDW06] und [Abe+18] wird ein Experiment durchgeführt, in dem Testpersonen unbekannte Lieder bewerten. Die Testpersonen werden in zwei Gruppen aufgeteilt. Eine *unabhängige* Gruppe und eine unter *sozialem Einfluss*. Testpersonen der Gruppe mit sozialem Einfluss werden zusätzlich in acht Welten eingeteilt und bekommen die Lieder nach der Downloadzahl sortiert und mit angezeigter Downloadzahl zu sehen, während den Personen aus der unabhängigen Gruppe jeweils eine zufällige Liste ohne Information über die Downloadzahl angezeigt wird. Durch die Ergebnisse zeigen die Autor*innen, dass unter sozialem Einfluss Ungleichheit und Unvorhersehbarkeit bezüglich der Popularität der Lieder besteht.

Munchnik et al. findet in [MAT13] heraus, dass sozialer Einfluss Bewertungsdynamiken verzerrt und zur Bildung von Filterblasen führen kann. Negativer sozialer Einfluss kann durch die Gruppenintelligenz jedoch wieder ausgeglichen werden.

In [LH14] wurde ein Experiment durchgeführt, bei dem Proband*innen Posts anderen Proband*innen empfehlen können, welche schließlich nach 5 unterschiedlichen Bewertungsmetriken sortiert werden:

- | | |
|-------------------------|---|
| 1. Zufall | in zufälliger Reihenfolge |
| 2. Popularität | nach der Anzahl der Empfehlungen sortiert |
| 3. Aktivität | nach der Zeit des letzten Empfehlungen sortiert |
| 4. Fixierung | stets in gleicher fixierter Reihenfolge |
| 5. Fixierung invertiert | stets in umgekehrter fixierter Reihenfolge |

2. Verwandte Arbeiten

Die Bewertungsmetriken werden mit dem *Gini*-Koeffizienten ausgewertet. Dabei schneidet die Zufallsmetrik, gefolgt von der Aktivitätsmetrik am besten, die fixierten Metriken am schlechtesten ab.

Für Votingsysteme mit den Bewertungsmöglichkeiten Up- und Downvotes eines Beitrages wird in [Mil] der *Wilson*-Score zur Verrechnung der Up- und Downvotes vorgeschlagen. Miller zeigt, weshalb andere Methoden schlechter geeignet sind. Der *Wilson*-Score findet im Reddit-Best-Kommentarranking Anwendung. Zhang stellt in [Zha+11] intuitive Axiome vor, welche eine Verrechnung der Bewertungen erfüllen sollte. Auch der *Wilson*-Score wird auf die Axiome geprüft.

Die Autor*innen von [HL12] und [LH10] stellen ein stochastisches Modell auf, um die Popularität von Beiträgen auf der Plattform Digg¹ vorherzusagen. Modellparameter, wie die Aktivitätsverteilung von User*innen, wird durch einen Datensatz von Digg gefittet. Das Modell wird validiert und erfasst die Hauptkomponenten der Bewertungsdynamiken von Digg.

[Sto15] untersucht die Korrelation zwischen Qualität und der Anzahl und Art der Bewertungen von Posts auf Hacker News und Reddit. Die Qualität eines Posts wird als die Bewertung beschrieben, die ein Post unter absolut fairen Bedingungen erhalten würde. Stoddard entwickelt eine Methode, um die Qualität von Posts auf den Plattformen abzuschätzen.

Kaltenbrunner et al. versucht in [KGL07] die User*inneninteraktionen, die ein Post auf der Plattform Slashdot² insgesamt erzeugt, auf Grundlage der erzeugten Interaktion in den ersten Minuten bzw. Stunden nach Veröffentlichung des Posts vorherzusagen.

In [WH08] wird gezeigt, dass die Popularität von Beiträgen Einfluss auf das Bewertungsverhalten hat. So sammeln populäre, gut bewertete Beiträge, zunehmend auch schlechte Bewertungen.

In [DN11] werden Kommentarsysteme von Nachrichtenagenturen auf die Wirkung von Beiträgen mit niedriger Qualität auf Nutzer*innen und Journalist*innen untersucht. Es wird gezeigt, wie die individuelle Lesemotivation Einfluss auf die Qualitätswahrnehmung hat. Um die Qualität zu verbessern, werden unter anderem Moderations- und Markierungsmethoden vorgeschlagen.

Luu schlägt in [Luu] vor, etwas zufälligen Lärm in die Bewertungsmetrik von Hacker News einzufügen, um die scharfe Aufmerksamkeitskante zwischen Posts, die auf der Startseite und denen, die auf den hinteren Seiten landen, auszuglätteten.

¹<https://digg.com/>

²<https://slashdot.org/>

3. Modell von Votingsystemen

In dem für diese Arbeit entwickeltem Model von Votingsystemen können User*innen Posts veröffentlichen. Diese sind für alle anderen User*innen sichtbar und können bewertet werden. Die Posts werden durch eine Bewertungsmetrik, unter Betrachtung verschiedener Postparameter, wie der Anzahl und Art der Bewertungen oder dem Zeitpunkt der Veröffentlichung, bewertet. Auf der Plattform werden die Posts absteigend nach ihrer Bewertung sortiert und in einer vertikalen Liste angezeigt. Je weiter ein*e User*in auf der Seite herunter scrollt, desto mehr Posts werden angezeigt. Die Posts sind nicht auf Seiten aufgeteilt, sondern befinden sich in einer kontinuierlichen Liste. Die Liste der Posts ist für alle User*innen, die die Plattform zum gleichen Zeitpunkt besuchen identisch und nicht personalisiert.

Im Laufe der Modellierung erhält die Plattform keine neuen User*innen, es werden jedoch neue Posts erstellt und hinzugefügt. Das agent*innenbasierte Modell wird für eine festgelegte Anzahl Iterationsschritte M_N simuliert. Für jeden Iterationsschritt i_M werden bestimmte Aktionen durchgeführt. Die User*innen interagieren in jedem Schritt mit dem Modell.

In diesem Kapitel werden die in der Simulation benötigten Modellparameter beschrieben.

3.1. Definition der Qualität

Posts in sozialen Medien besitzen eine Qualität, die von User*innen wahrgenommen wird. Die Postqualität besitzt mehrere Merkmale, wie zum Beispiel den Informationsgehalt, die Originalität oder die Richtigkeit der Grammatik und Rechtschreibung. User*innen nehmen die bestimmten Qualitätsmerkmale unterschiedlich wahr. Manchen User*innen erachten den Informationsgehalt eines Posts als wichtig, während andere eher auf die Originalität eines Beitrages achten.

Welche und wie viele Qualitätsmerkmale in der Realität existieren, kann nicht ermittelt werden, daher wird die Qualität künstlich durch Vektoren der Dimension Q_N modelliert. Jeder Eintrag eines Qualitätsvektors beschreibt die Ausprägung der Qualität in einem Qualitätsmerkmal. Ein Qualitätsvektor der Dimension Q_N beschreibt somit ein Objekt mit Q_N Qualitätsmerkmalen.

In einem Modell besitzen sämtliche Qualitätsvektoren die gleiche Dimension. Sie werden über eine kontinuierliche Q_N -dimensionale Verteilung Q erzeugt. Es bietet sich die Verwendung einer Q_N -dimensionalen Normalverteilung an. Einzelne Qualitätsmerkmale können so einfach in Korrelation gesetzt werden.

3.2. Bewertungsraum des Votingsystems

Ein Votingsystem kann einen Bewertungsraum V mit beliebiger Dimension V_N zur Verfügung stellen um Posts von User*innen bewerten zu lassen. Sind nur Upvotes, wie bei Hacker News, erlaubt, so handelt es sich um einen eindimensionalen Bewertungsraum. Auf Reddit sind auch Downvotes erlaubt, ein Bewertungsraum mit $V_N = 2$. Viele Onlineshops lassen in einem fünfdimensionalem Raum Artikel mit 1 bis 5 Sternen bewerten.

Diese Arbeit ist beschränkt auf $V_N = \{1, 2\}$, wobei bei $V_N = 1$ Upvotes, und bei $V_N = 2$ Up- und Downvotes betrachtet werden.

3.3. Posts

Posts besitzen beobachtbare Parameter, welche durch die User*inneninteraktion mit der Plattform entstehen und verändert werden. Diese Parameter sind auf einer produktiven Social Media Plattform üblicherweise in einer Datenbank gespeichert. Außerdem besitzen Posts nicht beobachtbare Parameter, welche von der Umwelt abhängig sind und artifiziell durch die Modellierung erzeugt werden.

3.3.1. Beobachtbare Parameter

Veröffentlichungszeitpunkt Der Veröffentlichungszeitpunkt $t_P = i_M$ ist der Modellschritt i_M , in welchem der Post veröffentlicht wurde.

Bewertungsvektor Der Bewertungsvektor b_P eines Posts besitzt die Dimension V_N . In ihm wird Bewertung der User*innen gespeichert. Für den zweidimensionalen Fall $V_N = 2$ ist:

$$b_P = \begin{pmatrix} n_{\uparrow P} \\ n_{\downarrow P} \end{pmatrix} \quad (3.1)$$

Dabei ist $n_{\uparrow P}$ die Anzahl der Upvotes und $n_{\downarrow P}$ die Anzahl der Downvotes.

Score Der Score s_P eines Posts enthält den Wert, welcher dem Post durch eine Bewertungsmetrik (Abschnitt 5) zugeordnet wurde.

Der Initialscore ι_0 , den die Posts zu Beginn ihrer Lebenszeit erhalten, kann variiert werden.

Betrachtungen Die Anzahl v_P der User*innen die den Post gesehen haben.

3.3.2. Nicht beobachtbare Parameter

Qualität Der Q_N -dimensionale Qualitätsvektor q_P beschreibt die Qualität eines Posts. Er wird aus der Qualitätsverteilung Q erzeugt.

Je größer einzelne Einträge in q_P sind, desto besser ist die Qualität eines Postes in diesem Qualitätsmerkmal.

Relevanz Die Relevanz gibt an, wie interessant ein Post ist. Relevante Posts sollen auf einer Kommunikationsplattform weit oben angezeigt werden.

Die Relevanz eines Posts hängt von der Qualität ab und kann sich aber mit der Zeit ändern. Auf einer News-Plattform spielt Aktualität eine wichtige Rolle. Posts, die gerade erst veröffentlicht wurden besitzen meist eine höhere Relevanz als Posts, die vor langer Zeit veröffentlicht wurden. Auf einer Q&A-Seite verlieren Beiträge mit verstreichender Zeit weniger an Relevanz. Die beste Antwort auf eine Frage bleibt dies meist auch für längere Zeit.

Die Relevanz r_P eines Posts in Formel 3.2 ergibt sich somit durch die Summe aller Einträge von q_P dividiert durch das in Relevanzgravität R_γ gesetzte Alter des Posts. Durch die Relevanzgravität wird festgelegt wie schnell die Posts an Relevanz verlieren. Die Relevanz ist abhängig vom Modellschritt i_M :

$$r_P = \frac{1}{(i_M - t_p)^{R_\gamma}} \sum_{i=1}^{Q_N} q_{P_i} \quad (3.2)$$

Für eine Q&A-Seite ist $R_\gamma = 0$ anzunehmen, da so die Relevanz über die Zeit konstant bleibt, während für eine News-Plattform $R_\gamma = 2$ gewählt werden kann.

3.4. Bewertungsmetriken

Bewertungsmetriken sollen Posts anhand ihrer Relevanz sortieren. Die Relevanz ist jedoch ein nicht beobachtbarer Parameter. Zur Approximation der Relevanz können nur beobachtbare Postparameter verwendet werden.

Das Kapitel 5 beschäftigt sich ausführlich mit Bewertungsmetriken.

3.5. User*innen

Um ein agent*innenbasiertes Modell einer Onlinekommunikationsplattform zu entwickeln ist es elementar das User*innenverhalten zu beschreiben. Die Parameter werden artifiziell durch das Modell definiert.

Aktivität Die Aktivität a_U beschreibt, wie oft User*innen aktiv sind, die Kommunikationsplattform besuchen und Posts betrachten. Dabei sind weder alle User*innen durchgehend noch gleichzeitig aktiv.

Die Aktivität wird als Wahrscheinlichkeit modelliert. Sie beschreibt, wie wahrscheinlich es ist, dass ein*e User*in in einem Iterationsschritt aktiv ist. Die Aktivität wird über die User*innen als β -verteilt angenommen. Einige denkbare β -Verteilungen sind in Abbildung 3.1 zu sehen. Da sie auf das Intervall $[0, 1]$ beschränkt sind, sind β -Verteilungen zur Modellierung von Wahrscheinlichkeiten geeignet. Mit ihnen lässt sich eine große Menge an User*innen beschreiben, die wenig aktiv sind und eine kleine Menge, welche sehr aktiv ist.

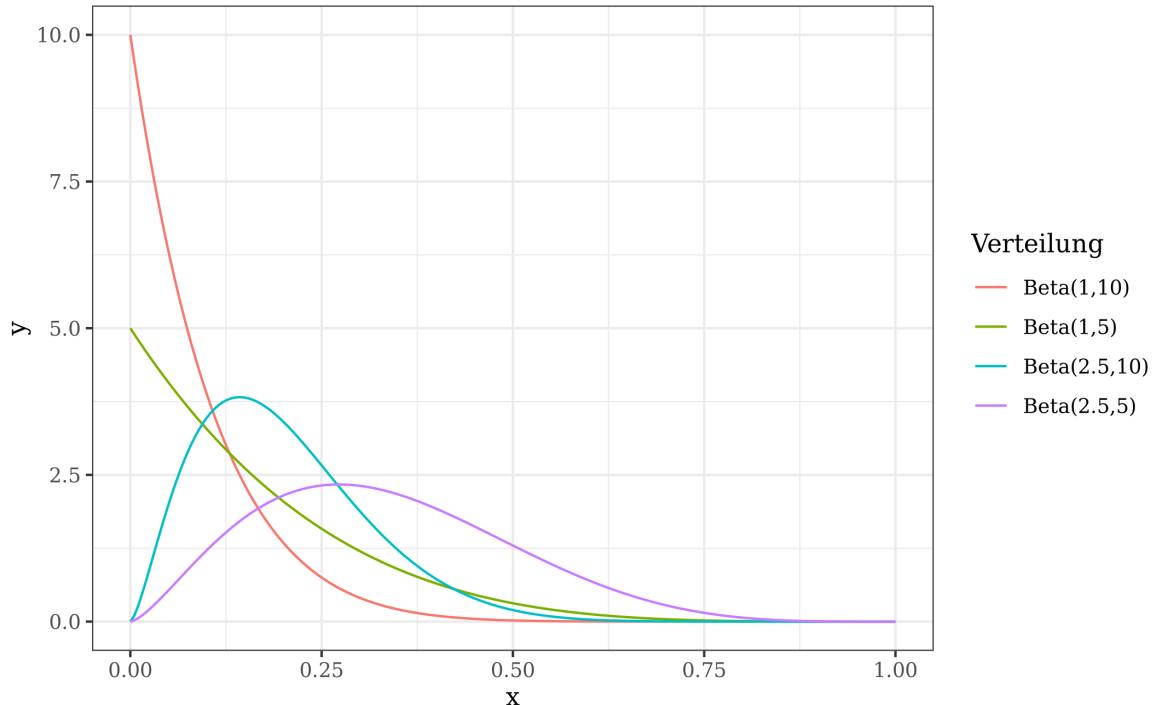


Abbildung 3.1.: Unterschiedliche β -Verteilungen zur Modellierung der User*innenaktivität

Konzentration Wenn ein*e User*in aktiv ist wird sie nicht alle Posts der Plattform anschauen. Einerseits weil die Plattform dazu zu viele Posts besitzt und andererseits, weil die Konzentration von User*innen auf der Plattform nicht unendlich ist. Wenn User*innen auf der Plattform aktiv sind, werden sie stets eine endliche Anzahl an Posts betrachten. Diese Anzahl wird durch die Konzentration definiert.

Die Konzentration k_U ist über die User*innen diskret-positiv verteilt. Es bietet sich die Modellierung mit einer Poisson-Verteilung an, da sich für diese ein diskreter Mittelwert definiert definiert wird, um welchen die Konzentrationswerte abweichen.

Qualitätsperzeption Durch die Qualitätsperzeption wird festgelegt wie User*innen Qualität wahrnehmen und wie stark ihre Qualitätswahrnehmung ausgeprägt ist. Die Qualitätsperzeption wird durch einen Vektor q_U mit Q_N Einträgen beschrieben. Jeder Eintrag beschreibt die Qualitätswahrnehmung in dem entsprechenden Qualitätsmerkmal. Je größer ein Eintrag im

Qualitätsvektor ist, desto höher ist die Fähigkeit der User*in die wahre Qualität des Posts in diesem Merkmal zu beurteilen.

q_U wird als Zufallswert aus der Qualitätsverteilung erzeugt.

Meinungsfunktion Wenn eine User*in U einen Post P betrachtet, bildet sie sich über den Post anhand der Postqualität und ihrer eigenen Qualitätsperzeption eine Meinung. Die Meinungsbildung wird durch die User*innenmeinungsfunktion modelliert. Durch diese Funktion $O(P, U) : \mathbb{R}^{Q_N} \rightarrow \mathbb{R}$ werden die beiden Q_N -dimensionalen Vektoren der Postqualität und Qualitätsperzeption auf einen skalaren Meinungswert $m_{P,U} = O(P, U)$ reduziert. Dieser drückt aus, wie gut die User*in den betrachteten Post empfindet. Je größer der Meinungswert ist, desto besser empfindet die User*in den Post.

Weitere Überlegungen zur User*innenmeinungsfunktion befinden sich Kapitel 4.

Meinungsverteilung Die Verteilung der Meinungswerte W wird zur Hilfe gezogen, um zu entscheiden ob Posts von User*innen bewertet werden. (Siehe dazu Abschnitt 3.5).

Zur Berechnung der empirischen Verteilungsfunktion von W werden n Posts und User*innen gebildet. Für jede User*in U_i wird mit jedem Post P_j der Meinungswert m_{P_j, U_i} gebildet. In 3.3 ergibt sich die empirische Verteilungsfunktion der verteilten Variable x :

$$F_W(x) = \frac{1}{n^2} \sum_{i,j=1}^n 1_{\{x \leq m_{P_j, U_i}\}} \quad (3.3)$$

Dabei ist $1_{\{x \leq O(P_i, U_j)\}} = 1$ für $x \leq O(P_i, U_j)$ und sonst 0.

Bewertungszufriedenheit Es wird angenommen, dass User*innen einen Post nur bewerten, wenn sie ausreichend (un)zufrieden mit diesem sind. Der berechnete Meinungswert muss dazu kleiner bzw. größer als ein bestimmtes Quantil der empirischen Dichteverteilung von W sein. Dieses Q_W -Quantil wird mit der Bewertungszufriedenheit z_U einer User*in definiert und hängt von der Dimension V_N des Bewertungsraumes ab. Für einen eindimensionalen Bewertungsraum ist $Q_W = 1 - z_U$. Für einen zweidimensionalen Bewertungsraum ist $Q1_W = \frac{z_U}{2}$ und $Q2_W = 1 - \frac{z_U}{2}$.

In Abbildung 3.2 sind beispielhafte Meinungsverteilungen in schwarz dargestellt. Die Quantile sind für eine User*in U mit $z_U = 0.1$ eingezeichnet. Auf der linken Seite ist Q_W für $V_N = 1$, auf der rechten Seite sind $Q1_W$ und $Q2_W$ für $V_N = 2$ rot auf der y -Achse eingezeichnet. Die Q_W -Quantile x_{Q_W} sind auf der x -Achse eingezeichnet. Um eine Bewertung eines Posts P von U hervorzurufen muss der Meinungswert $m_{P,U}$ in einen der rot markierten Bereiche fallen. Im Fall $V_N = 1$ muss $m_{P,U} > x_{Q_W}$ sein. Für $V_N = 2$, muss $m_{P,U} < x_{Q1_W}$ für eine negative Bewertung und $m_{P,U} > x_{Q2_W}$ für eine positive Bewertung sein.

Die Bewertungszufriedenheit ist kontinuierlich auf dem Intervall $[0, 1]$ verteilt. Zur Modellierung eignen sich wieder β -Verteilungen, da so modelliert werden kann, dass viele User*innen wenig Posts und wenige User*innen viele Posts bewerten.

3. Modell von Votingsystemen

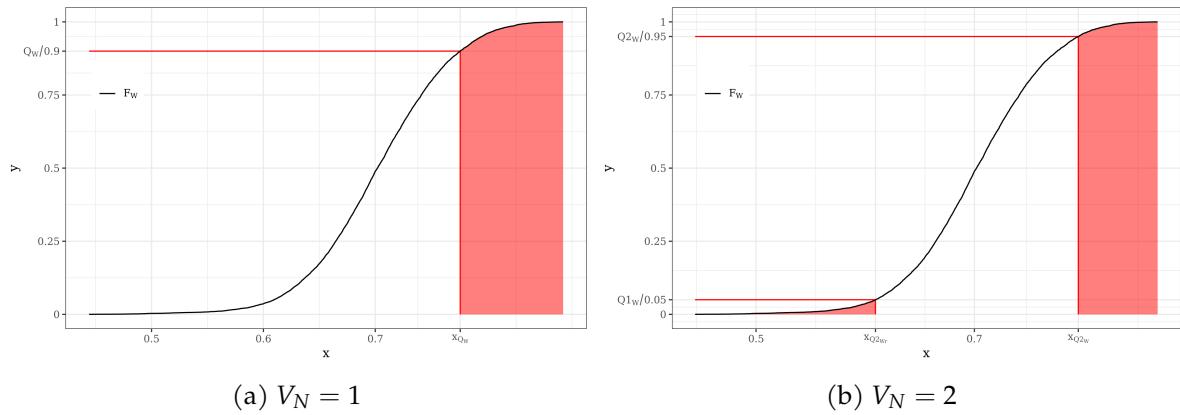


Abbildung 3.2.: Um eine Bewertung des Posts auszulösen, muss der Meinungswert einer User*in in den roten Bereich fallen

3.6. User*innen- und Postanzahl

In der folgenden Tabelle sind die Modellparameter der User*innen- und Postanzahl angegeben, sämtliche dieser Parameter können variiert werden:

U_N	Anzahl der User*innen
P_{startN}	Anzahl der Posts zu Beginn der Simulation
P_{iterN}	Anzahl der neuen Posts pro Iteration

4. User*innenmeinung

Für ein*e User*in U die einen Post P betrachtet ist die Meinungsfunktion $O(P, U) : \mathbb{R}^{N_Q} \rightarrow \mathbb{R}$, welche die beiden N_Q -dimensionalen Vektoren der Postqualität und Qualitätsperzeption auf den skalaren Meinungswert $m_{P,U} = O(P, U)$ reduziert. Dieser drückt aus wie gut die User*in den betrachteten Post empfindet. Dabei soll gelten:

1. $m_{P,U} \in [0, 1]$
2. Bei $m_{P,U} = 1$ wird der Post von der User*in als maximal gut empfunden
3. Bei $m_{P,U} = 0$ wird der Post von der User*in als maximal schlecht empfunden

4.1. Transformation der Qualitätsparameter

Um die Intervallgrenzen der genannten Kriterien zu erfüllen, wird eine Transformation der Qualitätsparameter vollzogen, die diese auf das Intervall $[0, 1]$ begrenzen. Die Transformation wird durch die logistische Funktion l in Formel 4.1 vollzogen:

$$l(x) = \frac{1}{1 + e^{-\frac{1}{2}x}} \quad (4.1)$$

Somit ergeben sich die transformierten Qualitäts- bzw. Qualitätsperzeptionsvektoren der Posts und User*innen durch die komponentenweise Anwendung von l auf q_P und q_U

$$\tilde{q}_P = l(q_P) \quad (4.2)$$

$$\tilde{q}_U = l(q_U) \quad (4.3)$$

4.2. Approximation der User*innenmeinungsfunktion

Die reale User*innenmeinungsfunktion ist hoch komplex und kann nur sehr vereinfacht modelliert werden. Im folgenden werden zwei Ansätze eingeführt, um die User*innenmeinungsfunktion $O(P, U)$ zu approximieren.

4. User*innenmeinung

4.2.1. Meinung im Konsens

Im Konsens wird davon ausgegangen, dass alle User*innen gute Posts mit hohen Qualitätsmerkmalwerten eher als gut empfinden und Posts mit niedrigen Qualitätsmerkmalwerten eher als schlecht wahrnehmen.

Der Konsens ist auf technischen und wissenschaftlichen Plattformen, zum Beispiel Hacker News, denkbar. Konstruktive Beiträge sind eher von destruktiven zu unterscheiden. Trotzdem können User*innen unterschiedlich viel Wissen in einzelnen Fachgebieten und dadurch einen höheren Anspruch an Posts in diesem Bereich haben.

In Formel 4.4 wird der Konsens ausgedrückt:

$$O_K(P, U) = \frac{1}{Q_N} \sum_{i=1}^{Q_N} \tilde{q}_P^{\tilde{q}_U} \quad (4.4)$$

Der transformierte Qualitätsvektor \tilde{q}_P des Posts wird komponentenweise mit der transformierten Qualitätsperzeption der User*in \tilde{q}_U exponiert und aufsummiert. Die Summe wird durch die Anzahl der Qualitätsmerkmale Q_N geteilt.

4.2.2. Meinung im Dissens

User*innen empfinden Posts als gut, die nah an ihrer eigenen Qualitätsperzeption liegen. Dadurch sind sich User*innen bei vielen Posts uneinig.

Der Dissens ist auf einer Diskussionsplattform denkbar. Dort können die Meinungen über Beiträge stark auseinander gehen. User*innen wollen Beiträge, welche ihre eigene Meinung wiedergeben für andere User*innen sichtbar machen.

In Formel 4.5 wird der Dissens mithilfe der euklidischen Distanz beschrieben. Es wird die euklidische Distanz aus \tilde{q}_P und \tilde{q}_U gebildet und durch $\sqrt{Q_N}$ geteilt. Dieser Term wird von 1 abgezogen. Eine User*in findet für $\tilde{q}_P = \tilde{q}_U$ einen Post maximal gut.

$$O_D(P, U) = 1 - \frac{\|\tilde{q}_P - \tilde{q}_U\|_2}{\sqrt{Q_N}} \quad (4.5)$$

4. User*innenmeinung

In Abbildung 4.1 werden anhand von jeweils zwei Beispielposts, $P1$ und $P2$, und -User*innen, $U1$ und $U2$, die beiden Meinungsfunktionen mit zwei Qualitätsmerkmalen x und y verglichen. Die Posts werden nach Farbe unterschieden, während die User*innen nach dem Symbol unterschieden werden. In der linken Abbildung sind die User*innen nach ihren transformierten Qualitätsmerkmalen dargestellt. Rechts ist der Meinungswert aus den beiden Meinungsfunktionen dargestellt. Herrscht Dissens, sind sich beide User*innen uneinig welchen Post sie besser finden. Im Konsens sind sich die User*innen einig, dass $P2$ besser ist. Trotzdem empfindet $U2$ den Post $P1$ deutlich schlechter als $U1$.

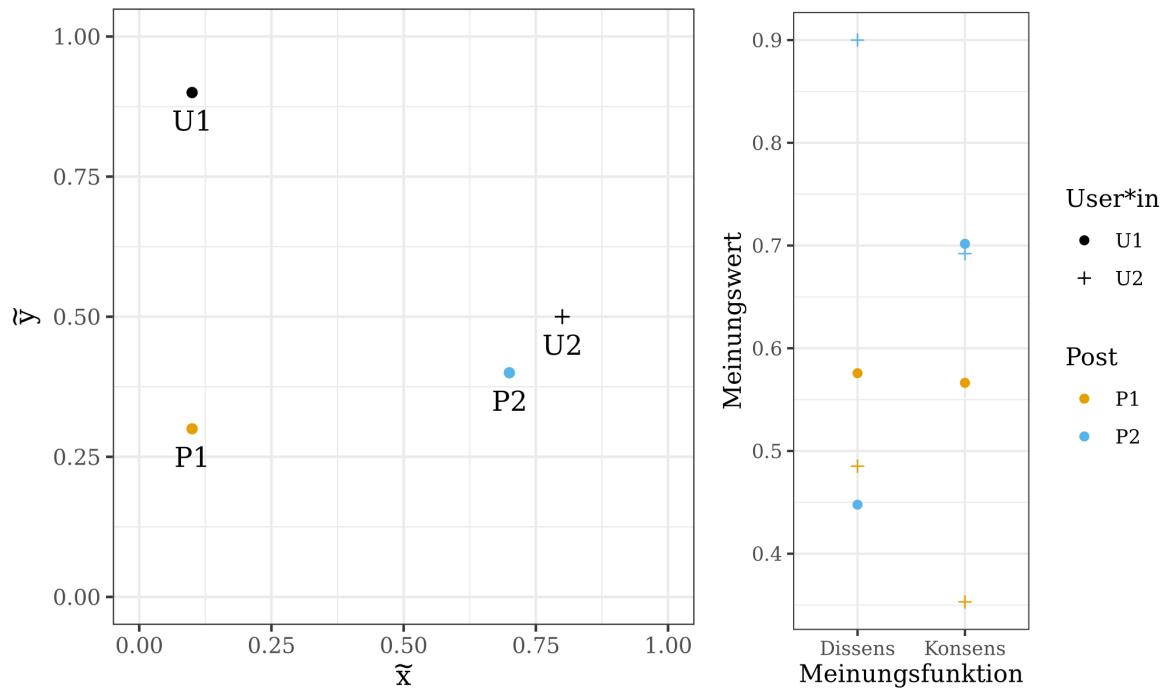


Abbildung 4.1.: Vergleich der beiden Meinungsfunktionen Konsens und Dissens

5. Bewertungsmetriken

Anhand von Bewertungsmetriken sollen Posts nach ihrer Relevanz sortiert werden. Ein Post P erhält durch eine Bewertungsmetrik den Score $s_P = B(P, i_M)$ zum Zeitpunkt i_M zugewiesen. Mit diesem zugewiesenen Score werden die Posts sortiert und auf der Plattform zur Verfügung gestellt.

Die nicht beobachtbaren Qualitätsparameter stehen in der Berechnung der Bewertungsmetrik nicht zur Verfügung, diese kann nur die beobachtbaren Parameter verwenden:

- a_P , Veröffentlichungszeitpunkt
- b_P , Bewertungsvektor
- s_P , Score
- v_P , Anzahl der Betrachtungen

In einer idealen Bewertungsmetrik B werden die Matthäus-Effekte reduziert, sie ist fair, wenn die folgenden Punkte erfüllt sind:

- für zwei Posts P_1 und P_2 mit $r_{P_1} > r_{P_2}$ ist $B(P_1, i_M) > B(P_2, i_M)$, zum Zeitpunkt i_M
- Für jeden Post P_i gilt $v_{P_i} > 1$, er wurde von mindestens einer*r User*in wahrgenommen

5.1. Bewertungstransformation

Die Bewertungsvektoren liegen im V_N -dimensionalen Raum vor. Innerhalb von Bewertungsmetriken werden die V_N dimensionale Bewertungsvektoren von Posts auf einen Skalar transformiert. Dies geschieht mit einer Bewertungstransformationsfunktion $\tau : \mathbb{R}^{V_N} \rightarrow \mathbb{R}$.

Für den zweidimensionalen Fall mit Up- und Downvotes ($n_{\uparrow P}, n_{\downarrow P}$) werden im folgenden einige Bewertungstransformationen vorgestellt.

5.1.1. Differenz

Es wird die Differenz aus Up- und Downvotes zu berechnet, so ergibt sich in Formel 5.1:

$$\tau_{diff}(b_P) = n_{\uparrow P} - n_{\downarrow P} \quad (5.1)$$

Die Differenz wird in der Reddit Hot Metrik standardmäßig verwendet.

5.1.2. Anteil

Berechnet wird der Anteil der Upvotes zur Gesamtzahl der Votes:

$$\tau_{anteil}(b_P) = \frac{n_{\uparrow P}}{n_{\uparrow P} + n_{\downarrow P}} \quad (5.2)$$

5.1.3. Wilson Score

Wie in [Mil] erläutert wird, ist die Verwendung der beiden vorherig vorgestellten Bewertungs transformationen in bestimmten Konstellationen von Up- und Downvotes nicht sinnvoll. Es wird die untere Grenze des Wilson-Konfidenzintervall für die Erfolgswahrscheinlichkeit p der Binomialverteilung p vorgeschlagen.

Mit der Gesamtzahl der Bewertungen eines Posts $n_P = n_{\uparrow P} + n_{\downarrow P}$, dem Punktschätzer $\hat{p}_P = \tau_{anteil}(b_P)$ und dem Quantil c_α der Normalverteilung zum Irrtumsniveau α ergibt sich in Formel 5.3 der Wilson Score:

$$\tau_{wilson}(b_P) = \frac{1}{1 + \frac{c_\alpha^2}{n_P}} \left(\hat{p}_P + \frac{c_\alpha^2}{2n_P} - c \sqrt{\frac{\hat{p}_P(1 - \hat{p}_P)}{n_P} + \frac{c_\alpha^2}{4n_P^2}} \right) \quad (5.3)$$

5.2. Bewertungsmetriken

Bewertungsmetriken berechnen den Score $B(P, i_M)$ für einen Post unter Betrachtung der Postparameter eines Postes P zum Zeitpunkt bzw. Modellschritt i_M .

5.2.1. Hacker News Metrik

In [Sal15a] wird die Hacker News Metrik beschrieben. Die Upvotes $n_{\uparrow P}$ eines Postes werden ins Verhältnis zum Alter $a_P = i_M - t_P$ gesetzt. Das Alter wird mit der Gravitationskonstante $\gamma = 1.8$ potenziert. Die Bewertungstransformation lautet $\tau(b_P) = n_{\uparrow P} - 1$, so ergibt sich die Hacker News Bewertungsmetrik in Formel 5.4:

$$B_{HN}(P, i_M) = \frac{n_{\uparrow P} - 1}{(a_P + 2)^\gamma} \quad (5.4)$$

5.2.2. Verallgemeinerte Hacker News Metrik

Das verallgemeinerte Hacker News Ranking verwendet eine beliebige Bewertungstransformation v und lässt sich somit auch auf höhere Bewertungsräume anwenden:

$$B_{vHN}(P, i_M) = \frac{\tau(b_P)}{(a_P + 2)^\gamma} \quad (5.5)$$

5.2.3. Reddit Hot Metrik

Die Reddit Hot Metrik wird in [Sal15b] beschrieben, hier wird ebenfalls die transformierte Bewertung τb_P eines Posts mit dem Alter verrechnet.

Der Zeitwert e_P wird als Differenz in Sekunden von diesem Zeitpunkt i_M zum Zeitpunkt E am 8.12.2005 um 07:46:43 in Formel 5.6 berechnet.

$$e_P = i_M - E \quad (5.6)$$

Die Bewertungstransformation der Reddit Hot Metrik ist standardmäßig die Differenz aus Up- und Downvotes, kann jedoch variiert werden:

$$\tau(b_P) = \tau_{diff}(b_P) \quad (5.7)$$

Der Parameter z wird in 5.8 auf das Maximum des Betrages von $\tau(b_P)$ aus Formel 5.7 und 1 gesetzt.

$$z_P = \begin{cases} |\tau(b_P)| & \text{falls } |\tau(b_P)| \geq 1 \\ 1 & \text{sonst} \end{cases} \quad (5.8)$$

Es ergibt sich die Reddit Hot Metrik in Formel 5.9:

$$B_R(P, i_M) = sign(\tau(b_P)) * log_{10}(z_P) + \frac{e_P}{4500} \quad (5.9)$$

Somit werden Posts mit fortschreitender Zeit nicht schlechter bewertet, wie bei der Bewertungsmetrik von Hacker News. Neuere Posts erhalten durch das Ansteigen von e_P eine höhere Bewertung, bei gleichem $\tau(b_P)$, als ältere Posts.

5.2.4. Viewmetrik

Die Viewmetrik basiert auf der in Abschnitt 5.2.2 beschriebenen Metrik. Es fließt die Anzahl der Betrachtungen des Posts v_P mit ein. Je weniger Betrachtungen für ein gleiches $\tau(b_P)$ benötigt sind, desto besser bewertet die Viewmetrik. Daraus ergibt sich mit dem gleichen a_P wie in Abschnitt 5.2.1 die Bewertungsmetrik in Formel 5.10:

$$B_V(P, i_M) = \frac{\frac{\tau_P(b_P)}{v_P+1}}{(a_P + 2)^\gamma} \quad (5.10)$$

5.2.5. Aktivitätsmetrik

Die Bewertung des Posts wird der Bewertung der letzten Iteration verrechnet und durch das Alter skaliert. Mit fortschreitender Zeit verlieren Posts in dieser Metrik an Score, es ergibt sich in Formel 5.11:

$$B_A(P, i_M) = \frac{\tau(b_P) - B_A(P, i_M - 1)}{(a_P + 2)^\gamma} \quad (5.11)$$

5.3. Zufallsbewertung

Nachdem Posts durch die Bewertungsmetrik bewertet wurden, kann der Score durch Zufall verunreinigt werden um den in [Luu] vorgeschlagenen Lärm zur Bewertung hinzuzufügen. In dieser Arbeit wurden zwei unterschiedliche Ansätze zur zufälligen Verunreinigung gewählt.

5.3.1. μ -Abweichung

Sei μ_s der Mittelwert der Scores aller Posts. Für einen Post P mit Score s_P wird die Verunreinigung d als Zufallszahl im Intervall $d_{\mu,P} \in [-|\mu_s - s_P|, |\mu_s - s_P|]$ definiert. So ergibt sich für den verunreinigten Score \tilde{s}_P des Post in Formel 5.12:

$$\tilde{s}_{\mu,P} = s_P + d_{\mu,P} \quad (5.12)$$

5.3.2. σ -Abweichung

Für die Standardabweichung der Scores aller Posts σ_s sei die Verunreinigung eine Zufallszahl im Intervall $d_{\sigma,P} \in [-\sigma_s, \sigma_s]$, sodass sich der verunreinigte Score eines Postes in Formel 5.13 ergibt:

$$\tilde{s}_{\sigma,P} = s_P + d_{\sigma,P} \quad (5.13)$$

6. Evaluationsmethode

Um die Modelle statistisch auszuwerten werden Evaluationsfunktionen erstellt. Evaluationsfunktionen können Modell- und User*innenparameter auswerten oder auch Funktionen auf ausgewerteten Parametern ausführen. In den Evaluationsmethoden stehen sämtliche, auch die beobachtbaren, Parameter zur Verfügung. Es gibt zwei unterschiedliche Arten von Evaluationsfunktionen:

- Iterationsevaluation:** Ausführung nach jeder Iteration des Modells
- Modellevaluation:** Ausführung nach jeder Modellsimulation

6.1. Iterationsevaluation

Iterationsevaluationsfunktionen werden nach jeder Iteration des Modells ausgeführt. Im folgenden werden einige Funktionen vorgestellt, welche Bewertungsmetriken nach unterschiedlichen Kriterien untersuchen.

6.1.1. Discounted Cumulative Gain

Der *Discounted Cumulative Gain*-Koeffizient (*DCG*) in [BGW18] dient zur Berechnung der Güte von Bewertungsmetriken. Der *DCG* misst, wie gut eine Bewertungsmetrik darin ist, relevante Posts für viele User*innen sichtbar zu machen. Er bestraft Bewertungsmetriken, welche Posts mit hoher Relevanz wenig Score gutschreiben. Die Posts werden nach ihrer Sortierung durchgegangen. Die standardisierte Relevanz eines Posts \tilde{r}_P wird durch den Logarithmus der Position in der Sortierung des Posts ins Verhältnis gesetzt. P_N ist die Anzahl der Posts:

$$DCG(B) = \sum_{i=1}^{P_N} \frac{2^{\tilde{r}_{P_i}} - 1}{\log_2(i + 1)} \quad (6.1)$$

Normalized Discounted Cumulative Gain (nDCG) Der *nDCG* normalisiert den *DCG* einer Bewertungsmetrik mit einer idealen Bewertungsmetrik B_I , welche die Posts absteigend nach Relevanz sortiert. $IDCG = DCG(B_I)$ ist der ideale *DCG*. Mit diesem ergibt sich:

$$nDCG(B) = \frac{DCG(B)}{IDCG} \quad (6.2)$$

Eine optimale Bewertungsmetrik erhält damit den Maximalwert $nDCG = 1$.

6.1.2. Gini-Koeffizient

In [LH14] und [SDW06] gibt der Gini-Koeffizient G an, wie gerecht Aufmerksamkeit der User*innen auf die Posts verteilt ist. Um die Aufmerksamkeit zu messen, wird die Anzahl der User*innen die einen Post v_p betrachtet haben verwendet. Je größer der Gini-Koeffizient ist, desto ungerechter ist die Verteilung der Views. Je größer G , desto stärker sind die Matthäus-Effekte in dieser Bewertungsmetrik ausgeprägt. Bei $G = 0$ ist die Aufmerksamkeit gerecht verteilt. Alle Posts wurden von der gleichen Anzahl User*innen gesehen. Wenn ein Post von allen User*innen gesehen wurde, alle anderen Posts hingegen von keine*r einzige*n User*in, ist $G \approx 1$ maximal. Der Gini-Koeffizient wird beschrieben durch:

$$G = \frac{1}{2P_N \sum_{i=1}^{P_N} v_{p_i}} \sum_{i,j} |v_{p_i} - v_{p_j}| \quad (6.3)$$

6.2. Modellevaluation

Modellevaluationsfunktionen werden nach der vollständigen Simulation eines Modells ausgeführt.

Modell- und User*innenparameter Da sich Modell- und User*innenparameter im Laufe der Simulation nicht ändern können, werden diese durch Modellevaluationsfunktionen ausgewertet.

6.2.1. Aggregation von Iterationsevaluationsfunktionen

Für ein Modell mit M_N Iterationsschritten können Modellevaluationsfunktionen über den von Iterationsevaluationsfunktionen erzeugten Datenvektor $x \in \mathbb{R}^{M_N}$ aggregieren, welcher für jede Iteration den Wert der Evaluationsfunktion enthält.

Die im vorherigen Abschnitt vorgestellten Iterationsevaluationsfunktionen werden durch die Trapezregel in Formel 6.4 aggregiert normalisiert:

$$T(x) = \frac{1}{M_N} \sum_{i=1}^{M_N} x_i - \frac{1}{2}(x_1 + x_{i_M}) \quad (6.4)$$

6.2.2. Posts ohne Betrachtungen

Der Prozentsatz an Posts, die von keine*r einzige*n User*in betrachtet wurden, werden durch die Funktion in Formel 6.5 gezählt. Je kleiner $P_{v=0}$, desto fairer ist die Metrik

$$P_{v=0} = \frac{1}{P_N} \sum_{N=1}^{P_N} 1_{v_{p_i}=0} \quad (6.5)$$

6. Evaluationsmethode

dabei ist $1_{v_{P_i}} = 1$, falls der Post P_i 0 Betrachtungen besitzt, sonst ist $1_{v_{P_i}} = 0$.

7. Implementierung

Die agentenbasierte Modell wurde mit der Bibliothek Agents.jl¹ für Julia² implementiert. Die verwendete Version von Agents.jl ist ein weitereintwickelter Fork der Verion 3.0.0.

Sämtliche in dieser Arbeit entwickelte Funktionalität ist in dem Paket VotingProtocols³ zusammengefasst.

7.1. agent*innenbasierte Modellierung von Votingsystemen

Die User*innen werden als Agent*innen des Modells modelliert. Posts, und die weiteren in Kapitel 3 beschriebenen Parameter, werden als Parameter des Modells von Agents.jl gespeichert. In der Implementierung stehen zwei Votingsysteme zur Verfügung:

upvote_system	erlaubt Upvotes	$V_N = 1$
up_and_downvote_system	erlaubt Up- und Downvotes	$V_N = 2$

Das Modell wird für eine festgelegte Anzahl an Iterationen berechnet. Der Ablauf einer Simulation ist in Algorithm 1 beschrieben. Das Modell wird mit U_N User*innen und P_{startN} Posts und sämtlichen weiteren Modellparametern initialisiert. In jedem Iterationsschritt wird für alle User*innen die definierte Agent*innenschrittfunktion (Abschnitt 7.1.1) ausgeführt. Diese legt das Bewertungsverhalten der User*innen fest und ist für jedes Votingsystem unterschiedlich. Nach der Berechnung der User*innenaktionen wird in jedem Iterationsschritt die Modellschrittfunktion (Abschnitt 7.1.2) ausgeführt und die definierten Iterationsevaluationsfunktionen (Abschnitt 7.3) aufgerufen. In dieser Funktion wird die Bewertungsmetrik angewendet und die Posts entsprechend angeordnet. Nach dem Durchlauf aller Iterationen werden schließlich die definierten Modellevaluationsfunktionen (Abschnitt 7.3) aufgerufen.

Algorithm 1 Modellsimulation (vereinfacht)

```
Erstelle Modell aus Modellkonfiguration
for all Iterationen do
    for all Agent*innen do
        Agent*innenschrittfunktion für Agent*in
    end for
    Modellschrittfunktion
    Rufe Iterationsevaluationsfunktionen auf
end for
Rufe Modellevaluationsfunktionen auf
```

¹<https://juliadynamics.github.io/Agents.jl/stable/>

²<https://julialang.org/>

³<https://github.com/melomys/voting-protocols>

7.1.1. Agent*innenschrittfunktion

Die Agent*innenschrittfunktion wird in Algorithm 2 beschrieben. Für ein*e User*in U wird zuerst berechnet, ob sie in der aktuellen Iteration aktiv ist. Dies wird anhand der Aktivitätswahrscheinlichkeit a_U berechnet. Ist U aktiv, werden so viele Posts von U auf der Plattform betrachtet, wie durch die Konzentration k_U vorgegeben sind. Für jeden Post P bildet sich U mit der User*innenmeinungsfunktion den Meinungswert $m_{P,U}$ und bewertet P , wenn die in Abschnitt 3.5 beschrieben Bedingungen erfüllt sind.

Algorithm 2 Agent*innenschritt

```

if User*in ist aktiv then
    for all Posts in Konzentrationsspanne do
        Betrachten und eventuell Bewerten des Posts
    end for
end if
```

7.1.2. Modellschrittfunktion

Der Ablauf der Modellschrittfunktion ist in Algorithm 3 dargestellt. Für jeden Post P wird mit der Bewertungsmetrik $s_P = B(P, i_M)$ berechnet. Anschließend werden $P_{N_{iter}}$ neue Posts dem Modell hinzugefügt. Falls es erwünscht ist, werden im nächsten Schritt die Postscores mit zufälligem Lärm verunreinigt. Nun können die Posts nach ihren Scores sortiert werden und die Modelliteration ist abgeschlossen.

Algorithm 3 Modellschritt

```

for all Posts do
    Postscore mit Bewertungsmetrik berechnen
end for
Neue Posts hinzufügen
if Mit zufälliger Abweichung then
    for all Posts do
        Postscore mit zufälliger Abweichung verunreinigen
    end for
end if
Posts nach Postscore sortieren
```

7.2. Erstellung von Modellkonfigurationen

Um unterschiedliche Modellkonfigurationen zu testen und zu vergleichen wurde in dieser Arbeit ein Framework entwickelt, das es ermöglicht, modular Modellparameter zu Modellkonfigurationen zusammenzustellen. Die Modellkonfigurationen werden über eine Liste von Tupeln in Julia definiert.

Eine bespielhafte Konfiguration ist im Listing 7.1 in der Liste `model_configs` zu sehen.

```

1 model_configs = [
2     (
3         up_and_downvote_system,
4         Dict(
5             :rating_metric => metric_reddit_hot,
6             :deviation_function => [
7                 no_deviation,
8                 std_deviation],
9             ),
10        ),
11        (
12            [upvote_system, up_and_downvote_system],
13            Dict(
14                :rating_metric => [
15                    metric_activation,
16                    metric_hacker_news],
17                :gravity => [0.5, 1.0, 1.5, 2.0],
18                ),
19            ),
20            (
21                :all_models,
22                Dict(
23                    :user_opinion_function => consensus
24                ),
25            ),
26        ],
27    ]

```

Listing 7.1.: Definition von Modellkonfigurationen in Julia

Der erste Eintrag der Tupel definiert das Votingsystem. Durch ein Votingsystem wird ebenfalls eine Grundkonfiguration A.1 definiert, diese stellt sicher, dass alle in der Simulation notwendigen Parameter gesetzt sind. In Zeile 3 wird das `up_and_downvote_system` ausgewählt. Im zweiten Eintrag können über ein Dictionary einzelne Modellparameter der Grundkonfiguration überschrieben werden. In den Zeilen 5 bis 8 werden einzelne Modellparameter überschrieben. Die Modellparameter werden als Symbole angesteuert. Der Modellparameter `deviation_function` wird mehrdeutig überschrieben, indem er durch eine Liste angegeben wird. Für jeden überschreibenden Wert wird eine eigene Modellkonfiguration angelegt. Das erste Tupel definiert somit zwei Modellkonfigurationen, die sich nur in `deviation_function` unterscheiden.

Der erste Eintrag des zweiten Tupels in Zeile 12 ist eine Liste aus Votingsystemen. Für jede der angegebenen Votingsysteme werden die im zweiten Eintrag des Tupels definierten

7. Implementierung

Modellparameter überschrieben. Die in Zeile 14 bis 17 angegeben Modellparameter werden mehrdeutig überschrieben. Der Parameter `rating_metric` wird doppelt definiert und `gravity` vierfach. Dadurch werden pro Votingsystem $2 * 4 = 8$ Modellkonfigurationen erstellt. Da zwei Votingsysteme angegeben sind, werden durch das zweite Tupel insgesamt $2 * (2 * 4) = 16$ Modellkonfigurationen festgelegt.

Das dritte Tupel in Zeile 21 besitzt als ersten Eintrag das Symbol `:all_models`. Damit wird festgelegt, dass die im zweiten Eintrag des Tupels überschriebenen Modellparameter auf alle definierten Modellkonfigurationen angewendet werden. Somit erhalten alle bereits definierten Modellkonfigurationen den Parameter `user_opinion_function` mit `consensus` zugewiesen.

Insgesamt wurden durch `model_config` 18 Modellkonfigurationen festgelegt. Zwei im ersten Tupel und 16 im zweiten Tupel.

7.3. Auswertung der Modelle

Die Modelle werden durch die Angabe von Evaluationsmethoden aus Kapitel 6 ausgewertet.

Die Iterations- und Modellevaluationsfunktionen werden über zwei Listen festgelegt. Die Evaluationsergebnisse werden analog in zwei Dataframes unter dem jeweiligen Namen der Evaluationsfunktion gespeichert.

Modellevaluationsmethoden haben Zugriff auf das berechnete Dataframe der Iterationsevaluationsmethoden.

Beispielhafte Konfigurationen der beiden Listen sind in Listing 7.2 zu sehen.

Die Iterationsevaluationsfunktionen `ndcg` und `gini` berechnen für jede Modelliteration den $nDCG$ bzw. Gini-Koeffizienten G der Bewertungsmetrik.

In der Liste `iter_eval_functions` sind die Iterationsevaluationsfunktionen festgelegt. Die Funktion `posts_with_no_views` berechnet $P_{v=0}$ des Models. Die Liste `model_eval_functions` definiert die Modellevaluationsparameter. Das Macro `@area_under` aggregiert die übergebenen Iterationsevaluationparameter mit der Trapezregel. Die durch das Macro erzeugte Funktion besitzt den Namen `area_under_<param>`. In diesem Beispiel wird so über alle Iterationsevaluationsparameter aggregiert. Mit `@model_parameter` können Modellparameter ausgewertet werden. Die erzeugte Funktion erhält den Namen des übergebenen Modellparameters. Hier werden der Initialscore der Posts, die Bewertungsmetrik, die Relevanzgravität und die User*innenmeinungsfunktion ausgewertet.

Simulation und Export Die Modellkonfigurationen und die Evaluationsfunktionen können der Funktion `export_data` übergeben werden. Diese führt die Berechnung und Evaluation des Modells aus und exportiert die Daten in das `rds`-Format Datenformat von R⁴. Die Datenanalyse kann so in R erfolgen.

⁴<https://www.r-project.org/>

```
1 iter_eval_functions =
2 [
3     ndcg,
4     gini,
5 ]
6
7 model_eval_functions =
8 [
9     posts_with_no_views,
10    @area_under(:ndcg),
11    @area_under(:gini),
12    @model_parameter(:init_score),
13    @model_parameter(:rating_metric),
14    @model_parameter(:relevance_gravity),
15    @model_parameter(:user_opinion_function),
16 ]
```

Listing 7.2.: Definition der Evaluationfunktionen

8. Simulationsergebnisse

In diesem Kapitel werden die Ergebnisse der Simulation der agent*innenbasierten Modelle vorgestellt.

Es werden die vier Bewertungsmetriken Verallgemeinertes Hacker News Metrik, Reddit Hot Metrik, Aktivitätsmetrik und Viewmetrik simuliert und verglichen.

Mit Blick auf die aufgestellten Kriterien einer fairen Bewertungsmetrik wird in der Auswertung der Koeffizient ρ in Formel 8.1 verwendet, welcher die relevanten aggregierten Evaluationsparameter $T(nDCG)$, den Gini-Koeffizienten $T(G)$ und den Anteil der nicht betrachteten Posts $P_{v=0}$ vereint. Der Koeffizient wird für optimale und faire Bewertungsmetriken minimiert und ist auf das Intervall $[0, 1]$ beschränkt.

$$\rho = \frac{1}{2} - \left(\frac{T(nDCG)}{2} - \frac{T(G)}{4} - \frac{P_{v=0}}{4} \right) \quad (8.1)$$

Mit der Modellkonfiguration Überblick A.2 werden eine Q&A- ($R_\gamma = 0$) und eine Newsplattform ($R_\gamma = 2$) mit jeweils der User*innenmeinungsfunktion Konsens O_K und Dissens O_D .

In Abbildung 8.1 sind die Simulationsergebnisse dargestellt. Jeder graue Punkt in der linken Abbildung beschreibt eine Modellsimulation, farblich hervorgehoben sind die Mittelwerte der Simulationsergebnisse jeder einzelnen Modellkonfiguration aus Überblick (s. Anhang A.2). Die Farbe gibt Aufschluss über die verwendete Bewertungsmetrik. Auf der x -Achse ist der aggregierte Gini-Koeffizient $T(G)$ aufgetragen, auf der y -Achse der aggregierte $nDCG$ $T(nDCG)$. Über die Größe der Punkte wird die Anzahl der Posts ohne Betrachtungen $P_{v=0}$ beschrieben. Durch den Korrelationsplot wird deutlich, dass die beiden News-Plattformen insbesondere und die Dissens-Q&A-Plattform stark durch den Spearman-Koeffizienten von ρ korreliert sind.

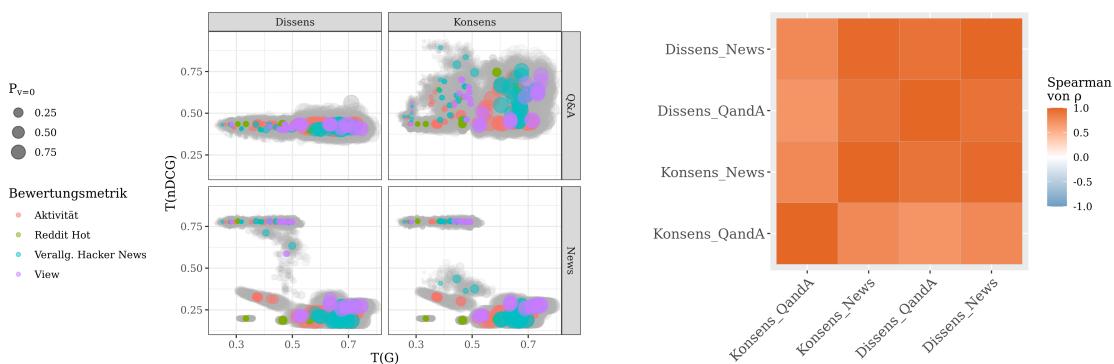


Abbildung 8.1.: Simulationsergebnisse nach vier Plattformarten. Die beiden News-Plattformen und die Q&A-Dissens-Plattform sind stark korreliert.

8. Simulationsergebnisse

Im Folgenden werden nur noch die beiden Plattformen auf denen Konsens herrscht betrachtet, die drei stark korrelierten Plattformen werden nicht einzeln betrachtet.

8.1. Größe des Bewertungsvektor

In Abbildung 8.2 ist wieder die Modellsimulation mit der Konfiguration Überblick zu sehen. Farblich markiert ist die Größe des Bewertungsvektors. Die Achsen sind identisch wie im vorherigen Plot. Für die Q&A-Plattform kann in 71.9% der Konfigurationen mit $V_N = 2$ ein besseres, kleineres ρ erzielt werden. Auf der News-Plattform nur in 65.6% der Konfigurationen. Bei a) werden die besten Ergebnisparameter mit $V_N = 1$ auf der News-Plattform erzeugt.

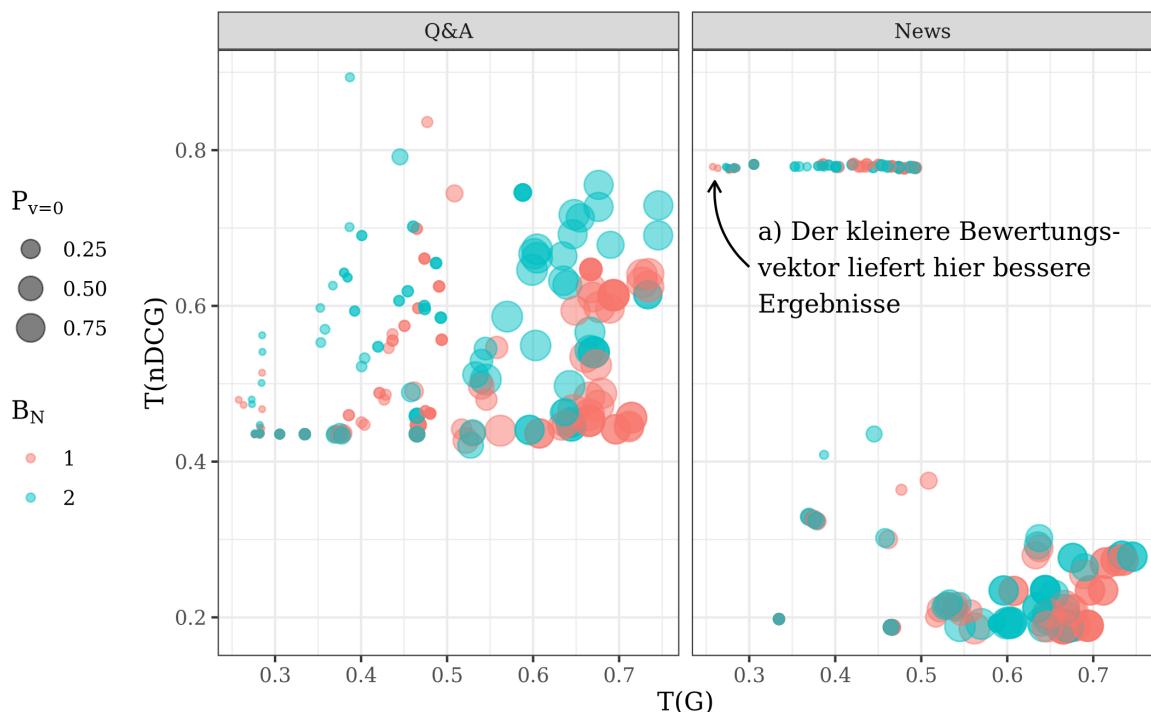


Abbildung 8.2.: In der Mehrheit der Konfigurationen ist der größere Bewertungsvektor überlegen.

8.2. Initialscore

Abbildung 8.3 zeigt ρ der Konfiguration Überblick der beiden Konsens-Plattformen. Auf der x-Achse ist der Initialscore ι_0 , auf der y-Achse der ρ -Koeffizient aufgetragen. Farblich markiert ist die verwendete Bewertungsmetrik. Für alle Metriken ist auf beiden Plattformen $\iota_0 = 0$ die schlechteste Wahl, da ρ am höchsten ausfällt.

Nach Konfiguration Initialscore (s. Anhang A.3) A.3 ist die Variierung des Initialscores ι_0 in den einzelnen Bewertungsmetriken in Abbildung 8.4 farblich dargestellt. Die Veränderung von ι_0 wirkt sich unterschiedlich auf die vier Metriken aus.

In der Konfiguration des verallgemeinerten Hacker News Metrik oben links wird durch die Erhöhung von ι_0 $T(G)$ und $P_{v=0}$ verringert und $T(nDCG)$ erhöht. Ab $\iota_0 > 70$ wird $T(nDCG)$

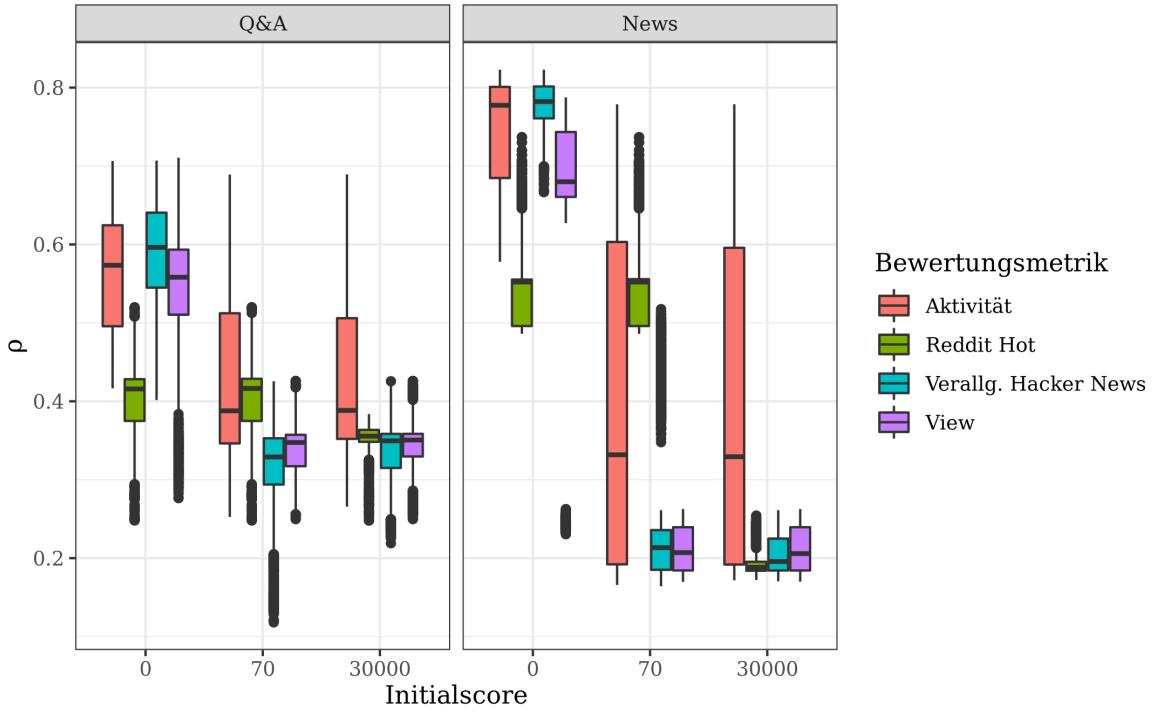


Abbildung 8.3.: Der Initialscore 0 ist für jede Metrik die schlechteste Wahl

wieder verringert, während $T(G)$ und $P_{v=0}$ konstant bleiben. Bei der Aktivitätsmetrik führt eine Erhöhung von ι_0 zur Abnahme von $P_{v=0}$, $T(nDCG)$ und $T(G)$. In der Viewmetrik wird zwischen $\iota_0 \in [10, 30]$ $T(G)$ und $P_{v=0}$ reduziert, ab $\iota_0 > 30$ wird hauptsächlich $T(nDCG)$ reduziert. Es findet ein ähnlicher Knick wie in der Hacker News Metrik statt, jedoch ab einem anderen ι_0 . Für die Reddit Hot Metrik ist $\iota_0 = \{0, 30000\}$. Für $\iota_0 = 30000$ erhalten neue Posts einen höheren Initialscore, als jemals von der Metrik zugewiesen wird. Der hohe Initialscore liefert bessere, kleinere $T(G)$ und $P_{v=0}$.

8.3. Gravität

Der Einfluss der Gravität der drei Bewertungsmetriken mit Gravität γ wird nach Konfiguration Parameter test A.4 in Abbildung 8.5 gezeigt. Die Aktivitätsmetrik besitzt für die Q&A-Plattform bei $\gamma = 0$ einen sehr schlechten Mittelwert von $P_{v=0} = 0.84$. Das beste ρ wird bei $\gamma = 0.5$ erzielt. Auf der Q&A-Plattform ist für die View- und Hacker News Metrik bei $\gamma = 0$ der ρ -Koeffizient am geringsten. Dieser steigt mit wachsendem γ für alle drei Metriken an. Auf der News-Plattform hat die Gravität keinen signifikanten Einfluss auf die Viewmetrik. Für die beiden weiteren Metriken wird über $\gamma \in [0, 2]$ der ρ signifikant verringert, ab $\gamma > 2$ steigt ρ wieder geringfügig an.

8.4. Bewertungstransformation

In Abbildung 8.6 ist die verwendete Bewertungstransformation in der Überblick-Konfiguration gekennzeichnet. Ein Datenpunkt beschreibt den Mittelwert einer Modellkonfiguration. Die

8. Simulationsergebnisse

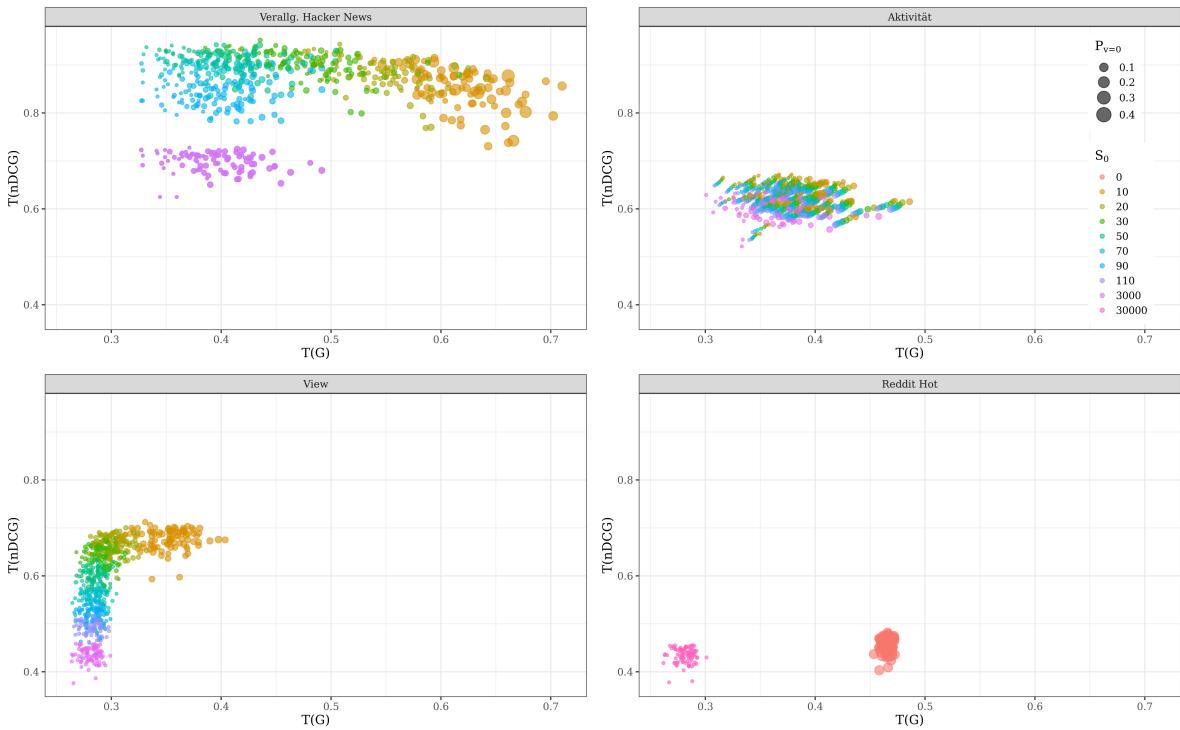


Abbildung 8.4.: Die Veränderung des Initialscores ι_0 wirkt sich unterschiedlich auf die vier Bewertungsmetriken aus.

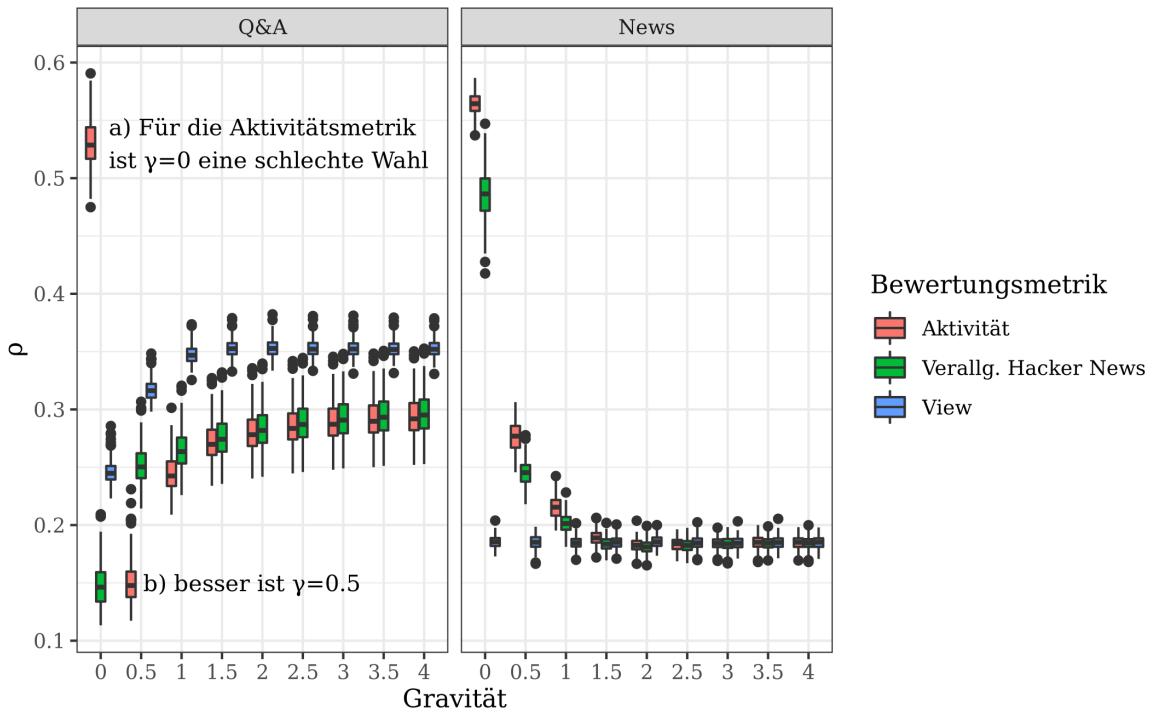


Abbildung 8.5.: Die Variierung der Gravität beeinflusst insbesondere auf die Aktivitäts- und Hacker News Metrik.

Bewertungstransformationen sind farblich markiert. Der linke Plot verwendet die üblichen Achsenbeschriftungen, im Boxplot sind die Bewertungsmetriken auf der x -Achse und ρ auf der y -Achse aufgetragen. Die verallgemeinerte Hacker News Metrik ist mit *VHN*, die Reddit Hot Metrik mit *RH* abgekürzt. Es zeigt sich, dass τ_{diff} in den meisten Konfigurationen eine

8. Simulationsergebnisse

größere Varianz bezüglich ρ als τ_{anteil} und τ_{wilson} aufweist. Auf der Q&A-Plattform besitzt τ_{diff} aber stets einen geringeren Mittelwert. Auf der News-Plattform ist die Performance von τ_{anteil} und τ_{wilson} sehr ähnlich und meist besser als τ_{diff} .

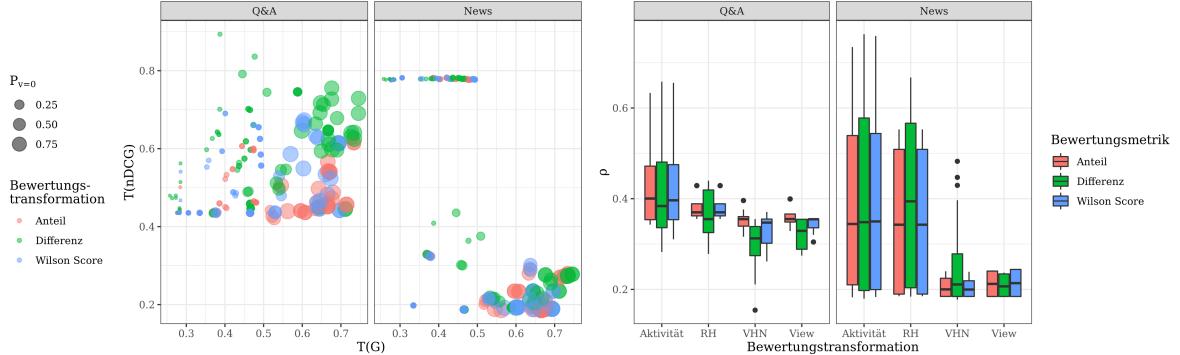


Abbildung 8.6.: Die Auswirkung der Bewertungstransformation. Auf einer Q&A-Plattform empfiehlt sich die Wahl von τ_{diff}

8.5. Zufallsabweichung

Die Boxplots in Abbildung 8.7 zeigen die Auswirkung der Zufallsabweichung in der Konfiguration Parameteretest (s. Anhang A.4) auf die einzelnen Evaluationsparameter für die vier Bewertungsmetriken. Auf einer Q&A-Plattform führt die Anwendung einer Zufallsabweichung bei allen Metriken außer der Hacker News Metrik zu einer signifikanten Reduzierung von $P_{v=0}$ und $T(G)$, es wird jedoch auch $T(nDCG)$ reduziert, durch die σ -Abweichung etwas mehr. Mit einer Zufallsabweichung wird $P_{v=0}$ und $T(G)$ für die Hacker News Metrik erhöht. Für eine News-Plattform wird für alle Metriken $T(G)$ und $P_{v=0}$ signifikant reduziert, während $T(nDCG)$ stabil bleibt.

8.6. Modell- und User*innenparameter

8.6.1. Iterationslänge

In Abbildung 8.8 links oben sind die Simulationsergebnisse nach der Konfiguration Parameteretest dargestellt. Die Iterationslänge M_N wird variiert und ist farblich markiert. Für die News-Plattform führt die Erhöhung von M_N zur einer Reduzierung von $T(nDCG)$ und $P_{v=0}$. Für die Q&A-Plattform sind mit den Boxplots die Parameter $T(nDCG)$, $P_{v=0}$ und $T(G)$ dargestellt. Auf der x-Achse ist M_N aufgetragen, durch die Farbe sind hier die Bewertungsmetriken gekennzeichnet. $T(nDCG)$ wird für die View- und Aktivitätsmetrik größer. Bei den drei restlichen Metriken wird $P_{v=0}$ und dessen Varianz verringert. Das Ansteigen der Iterationszahl hat nur geringen Einfluss auf $T(G)$, für die Hacker News Metrik wird $T(G)$ geringfügig größer. Die Varianz von $T(nDCG)$ verringert sich für alle Metriken. $P_{v=0}$ steigt für die Viewmetrik mit steigendem M_N . Außerdem steigt $T(nDCG)$ für die Aktivitäts- und Viewmetrik an, während

8. Simulationsergebnisse

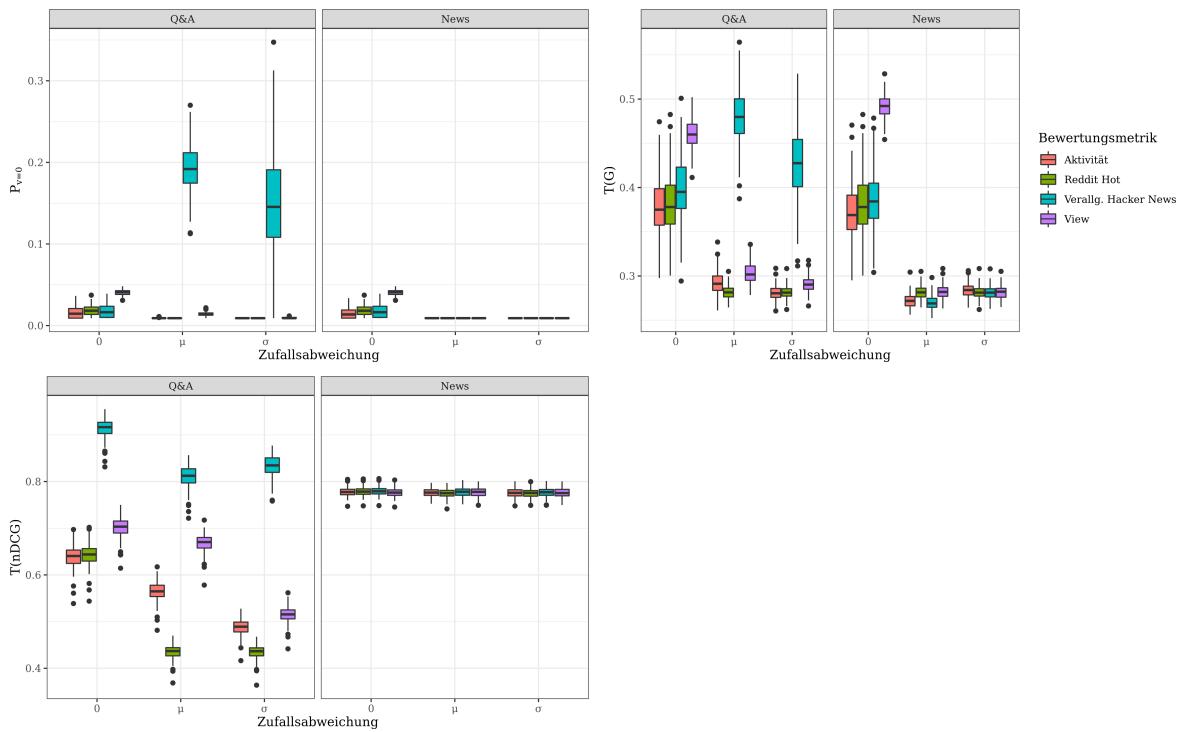


Abbildung 8.7.: Auf einer News-Plattform beeinflusst eine Zufallsabweichung die beobachteten Parameter positiv.

es für die anderen beiden Metriken keinen Anstieg erfährt. Die Aktivitäts- und Viewmetrik erhalten somit durch eine erhöhte Iterationslänge einen Fairnessvorteil, sie werden mit erhöhter Iterationslänge fairer, während es die anderen Metriken nicht tun.

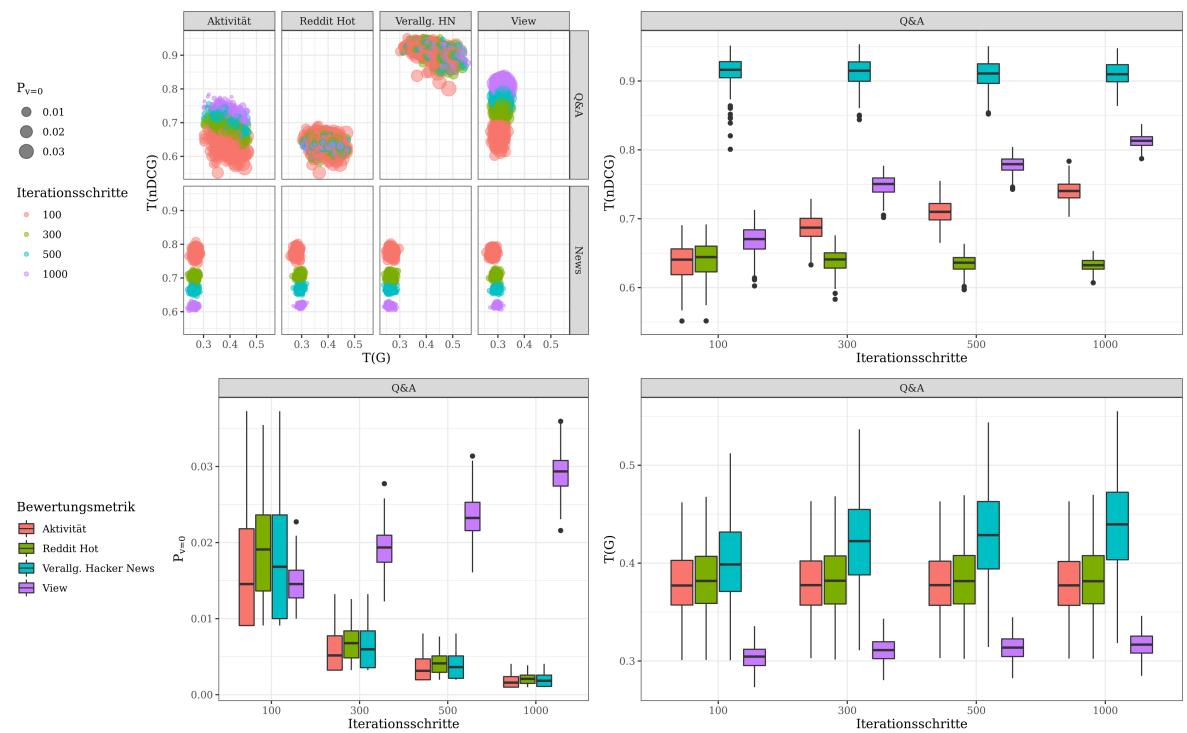


Abbildung 8.8.: Der Einfluss der Iterationslänge ist für die News-Plattform für alle Metriken gleich. Auf der Q&A-Plattform wirkt sich diese jedoch unterschiedlich aus.

8.6.2. Qualitätsraum

Der Einfluss der Größe des Qualitätsraums Q_N wird in Abbildung 8.9 nach den Ergebnissen der Konfiguration Parameter test untersucht. Im linken Plot ist die Größe des Qualitätsraums farblich gekennzeichnet. Durch eine Erhöhung von Q_N steigt $T(nDCG)$ auf der News-Plattform. Der Boxplot rechts zeigt, dass ρ durch die unterschiedlichen Q_N für die Q&A-Plattform für alle Bewertungsmetriken keine signifikante Änderung erfährt. Keine Bewertungsmetrik wird durch die bestimmte Wahl von Q_N bevorteilt.

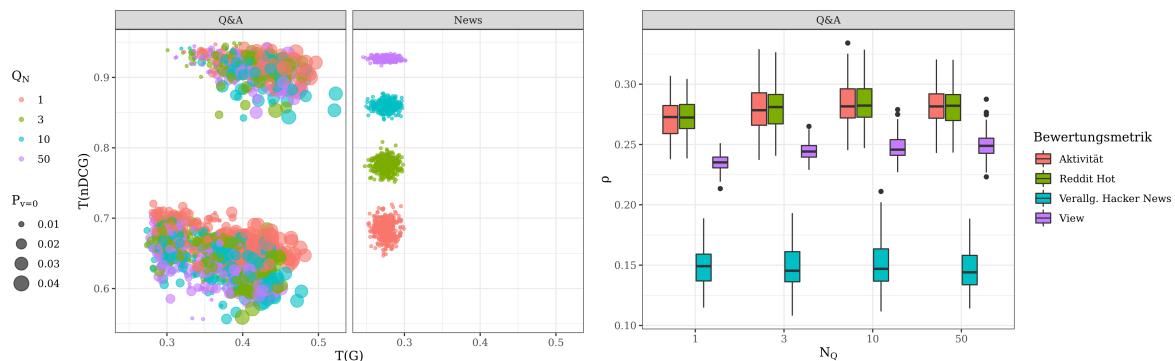


Abbildung 8.9.: Die Veränderung von Q_N wirkt sich nur auf die News-Plattform aus.

Wird die Qualitätsverteilung variiert, etwa durch Änderung des Erwartungsvektors oder der Kovarianzmatrix ergibt sich ein ähnliches Bild wie in Abbildung 8.9. Die Veränderung wirkt sich nur gleichmäßig auf die News-Plattform aus.

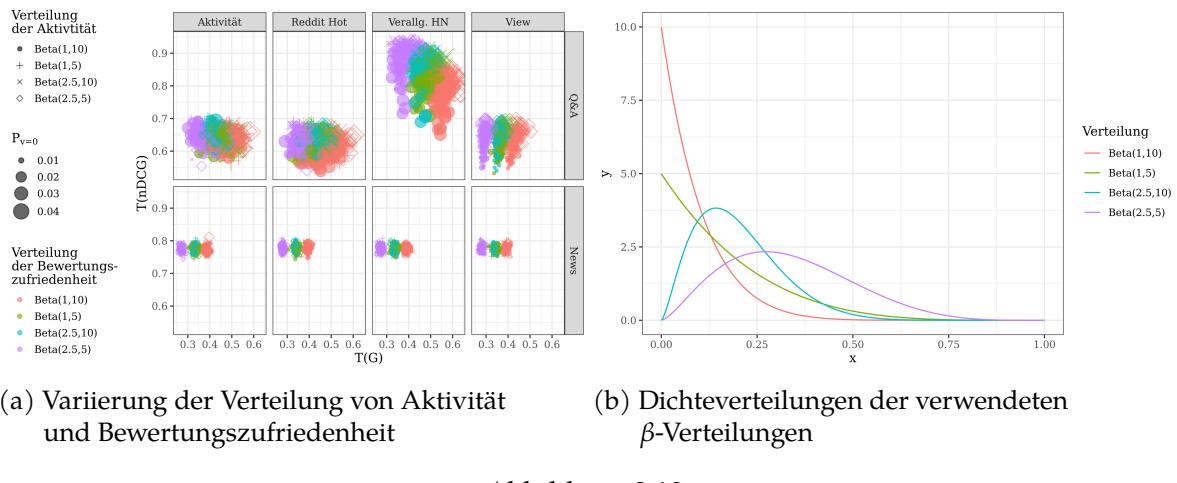
8.6.3. User*innenparameter

Mit variierten User*innenverteilungen sind Simulationsergebnisse nach der Parameter test-Konfiguration in Abbildung 8.10a dargestellt. Die Verteilungen der User*innenaktivität und der Bewertungszufriedenheit werden variiert und sind als unterschiedliche Symbole bzw. Farben im linken Plot für jede Bewertungsmetrik gekennzeichnet. In Abbildung 8.10b sind die Dichtefunktionen der vier verwendeten Verteilungen dargestellt. Wie im besonderen an der Viewmetrik auf der Q&A-Plattform zu sehen ist, führt die Variierung der Bewertungszufriedenheitsverteilung zur Verschiebung des Mittelwerts von $T(G)$, während die Variierung der Aktivitätsverteilung zur Verschiebung des Mittelwerts von $P_{v=0}$ und $T(nDCG)$ führt. Auf der News-Plattform hat die Variierung der Aktivitätsverteilung keinen signifikanten Einfluss. Die Simulationsergebnisse, die durch die Verwendung unterschiedlicher Verteilungen entstanden sind, stehen in starker Korrelation.

Durch die Variierung der Konzentration ergibt sich ein ähnliches Bild: wird der Erwartungswert der Konzentration erhöht, so verringern sich $T(G)$ und $P_{v=0}$.

Somit hat die Wahl der User*innenparameter keinen signifikanten Einfluss auf die Vergleichbarkeit von Modellkonfiguration, da keine einzelne Bewertungsmetrik von bestimmten User*innenparametern profitiert.

8. Simulationsergebnisse



8.6.4. User*innen- und Postanzahl

In Abbildung 8.11 ist der Einfluss der Anzahl der Startposts nach der Parametertest-Konfiguration dargestellt. Im Plot links ist farblich die Anzahl der Startposts P_{startN} gekennzeichnet. Wird die Anzahl der Startposts erhöht, werden die Evaluationsparameter schlechter. Je mehr Posts von Anfang an existieren, desto kleiner ist der Teil, den die User*innen zu Beginn erfassen können. Dadurch wird $T(G)$ und $P_{v=0}$ markant erhöht. Die Auswirkung auf $T(nDCG)$ fällt nicht so stark aus. Im Boxplot rechts ist auf der x -Achse die Anzahl der Startposts aufgetragen, farblich markiert ist die Bewertungsmetrik. Mit steigendem P_{startN} steigt ρ an. Die Ergebnisse mit unterschiedlicher Anzahl an Startposts sind stark miteinander korreliert. Die Bewertungsmetriken bleiben unabhängig von den Startposts vergleichbar, keine Bewertungsmetrik wird durch die bestimmte Wahl der Startposts bevorteilt.

Wird die Anzahl der neuen Posts pro Iteration oder die der User*innen variiert, sind die Ergebnisse der unterschiedlichen Anzahlen ebenfalls stark korreliert.

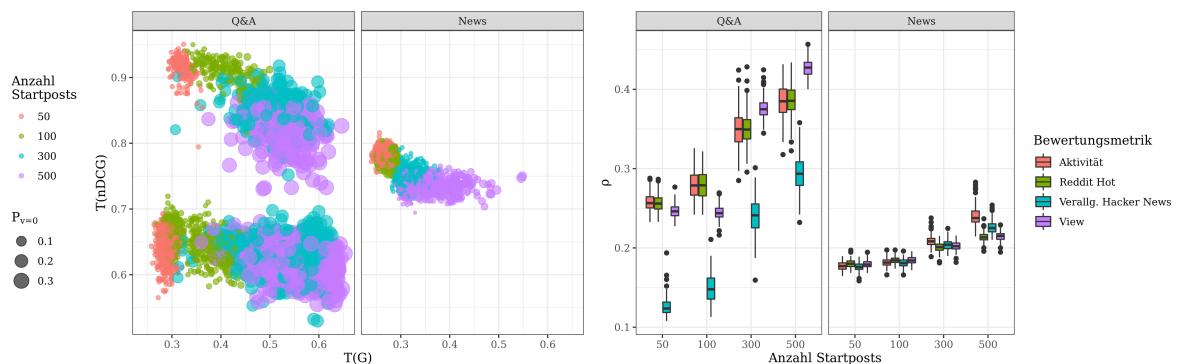


Abbildung 8.11.: Durch die Erhöhung der Anzahl von Startposts werden die Evaluationsparameter schlechter

9. Diskussion der Ergebnisse

Die unterschiedlichen Bewertungsmetriken wurden für eine Q&A-Platfoform mit $R_\gamma = 0$ und einer News-Plattform mit $R_\gamma = 2$, mit jeweils den User*innenmeinungsfunktionen Konsens und Dissens simuliert.

Bereits die Wahl des Votingsystems bzw. des Bewertungsraums hat Einfluss auf die Fairness einer Bewertungsmetrik. In dieser Simulation ist ein größerer Bewertungsraum von $V_N = 2$ in den meisten Fällen fairer. Jedoch gibt es insbesondere auf News-Plattformen einige Konfigurationen mit $V_N = 1$ welche besser abschneiden. Es lässt sich nicht mit Gewissheit sagen, welcher Bewertungsraum zur einer faireren Bewertungsmetrik führt, außerdem ist es interessant, größere V_N zu untersuchen.

Die in [Luu] vorgeschlagene zufällige Abweichung führt in dieser Simulation zur Verbesserung von Metriken auf News- und Dissens-Plattformen. Der aggregierte Gini-Koeffizient $T(G)$ und die Anzahl der Posts ohne Betrachtungen kann signifikant reduziert werden, während der aggregierte $nDCG$ $T(nDCG)$ stabil bleibt. Hier empfiehlt sich die Anwendung einer zufälligen Abweichung.

In [Mil] wird der Wilson-Score τ_{wilson} vorgeschlagen um Up- und Downvotes eines Posts zu evaluieren. In der Simulation zeigt sich, dass der Wilson Score nicht in allen Fällen der Differenz von Up- und Downvotes τ_{diff} überlegen ist. Auf einer Q&A-Plattform auf der Konsens herrscht ist die τ_{diff} überlegen.

Bewertungsmetriken liefern bessere Ergebnisse, wenn Posts mit einem Initialscore $\iota_0 > 0$ ausgestattet werden. Die Wahl des besten Initialscores ist von der Bewertungsmetrik abhängig. Bei dem Initialscore $\iota_0 = 0$ werden die neuen Posts unter allen bereits existierenden Posts angezeigt und erhalten somit nur sehr wenige Betrachtungen. Sobald $\iota_0 > 0$ gewählt wird, erhalten neue Posts von Beginn an Betrachtungen und gute Posts so die Chance gute Bewertungen zu erhalten. Für die Reddit Hot Metrik konnte aufgrund der Komplexität der Metrik der Initialscore nur marginal untersucht werden, es ist jedoch zu vermuten, dass dieser, wie bei den drei weiteren Metriken, weder kleiner noch größer als der Score sämtlicher Posts der Bewertungsmetrik zu wählen ist.

Die Wahl der Gravität hat einen großen Einfluss auf die Fairness von Bewertungsmetriken, diese sollte daher mit Bedacht gewählt werden. Während auf einer Q&A-Plattform kleine Gravitäten $\gamma = 0 + \epsilon, \epsilon \leq 0.5$ vorteilhaft sind, ist auf einer News-Plattform mit $R_\gamma = 2$ am besten $\gamma = 2$ zu wählen. Möglicherweise ist für $R_\gamma \approx \gamma$ die Bewertungsmetrik am fairesten.

Die Evaluationsparameter $T(G)$ und $P_{v=0}$ sind mit einem Spearman-Koeffizienten von 0.92 in der Überblick-Konfiguration sehr stark korreliert. Der Gini-Koeffizient achtet implizit auf die Anzahl der Posts ohne Betrachtungen, worauf diese hohe Korrelation zurückzuführen ist.

9. Diskussion der Ergebnisse

In dieser Arbeit wurde der sehr große Featureraum nicht vollständig exploriert. Einzelne Metrikparameter wie der Initialscore werden nur unter bestimmten Modellkonfigurationen simuliert und verglichen. Es ist möglich, dass Kombinationen von Metrikparametern sehr gute Ergebnisse liefern, zu diesen aber keine Simulation ausgeführt wurde und sie nicht entdeckt wurden. Mit Gewissheit kann nicht gesagt werden, welche Bewertungsmetrik unter einer bestimmten Konfiguration auf einer der Plattformarten am fairesten ist. Es zeigt sich jedoch, dass durch die Wahl der Bewertungsmetrik die Fairness der Sortierung von Posts beeinflusst und damit die faireste Bewertungsmetrik im Featureraum gefunden werden kann.

Einfluss der Modell- und User*innenparameter Die User*innenmeinungsfunktion kann, je nach Plattform die modelliert werden soll, als Konsens oder als Dissens gewählt werden. Wird eine News-Plattform modelliert, sind die Ergebnisse im Konsens und Dissens sehr stark korreliert. Die hohe Relevanzgravität auf der News-Plattform scheint die unterschiedlichen Meinungsfunktionen zu egalisieren. Außerdem betrachten die User*innenmeinungsfunktionen nur die Qualität eines Postes und nicht das Alter, dieses wird jedoch bei der Berechnung der Relevanz eines Posts bestraft. Es entsteht eine grundsätzliche Abweichung zwischen der Relevanz und Meinungsfunktionen von älteren Posts. Diese kann in einer zukünftigen Implementierung durch das Einfließen der Relevanzgravität in die User*innenmeinungsfunktion verhindert werden.

Die Erhöhung der Iterationslänge führt erwartungsgemäß zu einer Verringerung der Varianz und damit Präzisierung der betrachteten Ergebnisparameter. Im Fall von Q&A-Plattformen werden einzelne Bewertungsmetriken von hohen Iterationslängen bevorzugt, sie liefern dort bessere Ergebnisse. Daher ist es weiterhin interessant, die Bewertungsmetriken unter höheren Iterationslängen zu untersuchen und die Langzeitperformance zu analysieren.

Die Vergleichbarkeit der Ergebnisse ist nicht unmittelbar abhängig von der Wahl der User*innenparameter. Zwar können die Ergebnisse zum Beispiel durch die Wahl der Konzentrationsverteilung bezüglich des aggregierten Gini-Koeffizienten verschoben werden, jedoch erfahren dann sämtliche Modellkonfigurationen diese Verschiebung, sodass keine einzelnen Modellkonfigurationen durch die bestimmte Wahl der User*innenparameter bevorteilt wird.

Erfreulicherweise scheinen die Änderungen des Qualitätsraum ebenfalls keinen Einfluss auf die Vergleichbarkeit der Ergebnisse zu haben, keine Bewertungsmetrik wird durch die bestimmte Wahl eines Qualitätsraums bevorteilt. Der Qualitätsraum ist nicht bekannt und muss artifiziell erzeugt werden, ohne Einfluss auf die Vergleichbarkeit der Ergebnisse kann der Qualitätsraum sinnvoll beliebig gewählt werden.

Auch die Variierung der Post- und User*innenzahlen hat keinen gravierenden Einfluss auf die Vergleichbarkeit der Ergebnisse. Wie zu erwarten steigt mit zunehmender Postanzahl der Gini-Koeffizient und die Anzahl der Posts ohne Betrachtungen. Auf einer reellen Social Media Plattform ist die Anzahl der User*innen und Posts um ein Vielfaches größer als in dieser Simulation. Es ist sicherlich aufschlussreich, sich näher an die reellen Zahlen heranzutasten, um zu überprüfen, ob die Vergleichbarkeit der Ergebnisse erhalten bleibt, sodass in der Simulation weiterhin kleine User*innen- und Postanzahlen verwendet werden können.

10. Fazit

In dieser Arbeit wurde ein agent*innenbasiertes Modell zur Simulation einer Social Media Plattform entwickelt und in Julia implementiert. Auf der modellierten Plattform können User*innen mit Posts interagieren, indem sie diese bewerten. Das Nutzer*innenverhalten wird für Plattformen, auf denen Konsens und Dissens herrscht, modelliert. Es werden vier Bewertungsmetriken, darunter die Reddit Hot und die Hacker News Metrik, eingeführt. Für die Bewertungsmetriken wurde ein Fairnessbegriff definiert, anhand dessen sie vergleichbar werden. Die Bewertungsmetriken besitzen einige variable Parameter, wie den Initialscore, die Zufallsabweichung und die Bewertungstransformation. Mit einem in Julia entwickelten Framework können die Bewertungsmetriken und das agent*innenbasierte Modell konfiguriert und simuliert werden. Die Bewertungsmetriken werden mit Evaluationsparametern wie dem $nDCG$ oder dem Gini-Koeffizienten ausgewertet. Die Ergebnisse der Simulation zeigen, dass die Bewertungsmetriken sich in ihrer Fairness unterscheiden und die Wahl der fairen Bewertungsmetrik von der Art der Plattform abhängig ist. Die artifiziell erzeugten User*innenparameter besitzen nur einen geringen Einfluss auf die Simulationsergebnisse.

11. Ausblick

Die Simulationen wurden mit kleinen Iterationslängen, User*innen- und Postanzahlen durchgeführt. Einige Bewertungsmetriken werden durch eine höhere Iterationszahl besser, während andere im Ergebnis stabil bleiben. Wird die Iterationslänge, wie die User*innen- und Postanzahlen erhöht, werden die Ergebnisse genauer und ermöglichen besseren Aufschluss, welche Bewertungsmetrik am fairen ist. Es wäre interessant Simulationen mit höheren Anzahlen durchzuführen.

Die Implementierung ist beschränkt auf Votingsysteme mit $V_N = \{1, 2\}$, Modelle, die User*innen Posts nur positiv oder negativ bewerten können. Es zeigt sich, dass der zweidimensionale Bewertungsraum in mehr Konfigurationen bessere Ergebnisse liefert. Es wäre interessant, zu untersuchen wie sich Modelle mit größeren $V_N > 2$ verhalten.

Im Modell sind die Verteilungen, welche das User*innenverhalten definieren nicht korreliert. Dies ist in der Realität jedoch sicherlich der Fall. Nutzer*innen, die häufig aktiv sind, weisen sicherlich auch ein ausgeprägteres Bewertungsverhalten aus, sie bewerten Posts häufiger. Eine zukünftige Arbeit könnte die User*innenparameter in Korrelation setzen.

11. Ausblick

Wenn User*innen im Modell einen Post betrachten, entscheiden sie rein nach der wahrgenommenen Qualität, ob und wie sie den Post bewerten. Der soziale Einfluss, der in einem realen System durch die angezeigten Bewertungen entsteht, wird in diesem Modell nicht erfasst. Um dies zu realisieren, muss die Bewertungszufriedenheit durch den neuen sozialen Einfluss angepasst werden. Auch das Alter der Posts wird in den aktuellen User*innenmeinungsfunktionen nicht wahrgenommen, die User*innenmeinungsfunktionen könnten dahingehend angepasst werden.

Die Framework bietet die einfache Definition von neuen Bewertungsmetriken und deren Parameter. In [Die15] wird das *Dirichlet Smoothing* verwendet, um die Bewertungen der Posts zu transformieren, dies könnte ebenfalls implementiert und getestet werden. Aktuell können nur fixe Initialscores für Posts angegeben werden, es ist jedoch auch denkbar die Initialscores dynamisch zu berechnen. Neue Posts könnten zum Beispiel immer den Mittelwert der aktuellen Scores der Posts als Initialscore erhalten.

Der Featureraum für die Parameter der Bewertungsmetrik und des Votingsystems kann begrenzt werden, sodass ein Optimierungsproblem formuliert werden und zum Beispiel nach der Minimierung von ρ optimiert werden kann. So wäre es möglich für eine bestimmte Plattformart die optimale Bewertungsmetrik zu finden.

Literatur

- [Abe+18] A. Abeliuk u. a. „Measuring and Optimizing Cultural Markets“. In: (2018).
- [AT05] G. Adomavicius und A. Tuzhilin. „Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions“. In: Bd. 17. 6. 2005, S. 734–749.
- [BGW18] A.J. Biega, K.P. Gummadi und G. Weikum. „Equity of attention: Amortizing individual fairness in rankings“. In: *41st International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2018*. ACM, 2018, S. 405–414.
- [Bur+20] L. Burbach u. a. „Netlogo vs. Julia: Evaluating different options for the simulation of opinion dynamics“. In: Bd. 12199 LNCS. Springer, 2020, S. 3–19.
- [Cas19] C. Castillo. „Fairness and Transparency in Ranking“. In: *ACM SIGIR Forum* 52 (2019).
- [CSV18] L.E. Celis, D. Straszak und N.K. Vishnoi. „Ranking with fairness constraints“. In: *Leibniz International Proceedings in Informatics, LIPIcs*. Bd. 107. Schloss Dagstuhl-Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing, 2018.
- [Deg74] M.H. Degroot. „Reaching a consensus“. In: *Journal of the American Statistical Association* 69.345 (1974), S. 118–121.
- [Die15] F. Dietze. *Quality Assurance in a Structured Collaborative Discussion System*. 2015.
- [DN11] N. Diakopoulos und M. Naaman. „Towards quality discourse in online news comments“. In: *Proceedings of the ACM Conference on Computer Supported Cooperative Work, CSCW*. Hangzhou, 2011, S. 133–142.
- [FGR16] S. Flaxman, S. Goel und J.M. Rao. „Filter bubbles, echo chambers, and online news consumption“. In: Bd. 80. Specialissue1. 2016, S. 298–320.
- [HL12] T. Hogg und K. Lerman. „Social dynamics of digg“. In: *EPJ Data Science* 1.1 (2012), S. 1–26.
- [KGL07] A. Kaltenbrunner, V. Gómez und V. López. „Description and prediction of Slashdot activity“. In: *Proceedings - 2007 Latin American Web Conference, LA-WEB 2007*. Santiago, 2007, S. 57–66.
- [LH10] K. Lerman und T. Hogg. „Using a model of social dynamics to predict popularity of news“. In: *Proceedings of the 19th International Conference on World Wide Web, WWW '10*. Raleigh, NC, 2010, S. 621–630.
- [LH14] K. Lerman und T. Hogg. „Leveraging position bias to improve peer recommendation“. In: *PLoS ONE* 9.6 (2014).

- [Luu] D. Luu. *Why HN Should Use Randomized Algorithms*. <http://danluu.com/randomize-hn/>. Zugriff am 10.08.2020.
- [MAT13] L. Muchnik, S. Aral und S.J. Taylor. „Social influence bias: A randomized experiment“. In: *Science* 341.6146 (2013), S. 647–651.
- [Mil] E. Miller. *How Not To Sort By Average Rating*. <https://www.evanmiller.org/how-not-to-sort-by-average-rating.html>. Zugriff am 10.08.2020.
- [MVN19] L. Mastroeni, P. Vellucci und M. Naldi. „Agent-Based Models for Opinion Formation: A Bibliographic Survey“. In: *IEEE Access* 7 (2019), S. 58836–58848.
- [Sal15a] A. Salihefendic. *How Hacker News ranking algorithm works*. <https://medium.com/hacking-and-gonzo/how-hacker-news-ranking-algorithm-works-1d9b0cf2c08d>. Zugriff am 14.08.2020. 2015.
- [Sal15b] A. Salihefendic. *How Reddit ranking algorithms work*. <https://medium.com/hacking-and-gonzo/how-reddit-ranking-algorithms-work-ef111e33d0d9>. Zugriff am 14.08.2020. 2015.
- [Sar+01] B. Sarwar u. a. „Item-based collaborative filtering recommendation algorithms“. In: 2001, S. 285–295.
- [SDW06] M.J. Salganik, P.S. Dodds und D.J. Watts. „Experimental study of inequality and unpredictability in an artificial cultural market“. In: *Science* 311.5762 (2006), S. 854–856.
- [SJ18] A. Singh und T. Joachims. „Fairness of exposure in rankings“. In: Association for Computing Machinery, 2018, S. 2219–2228.
- [Sto15] G. Stoddard. „Popularity dynamics and intrinsic quality in reddit and hacker news“. In: *Proceedings of the 9th International Conference on Web and Social Media, ICWSM 2015*. AAAI Press, 2015, S. 416–425.
- [TL14] A. Tsang und K. Larson. „Opinion dynamics of skeptical agents“. In: *13th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2014*. Bd. 1. International Foundation for Autonomous Agents und Multiagent Systems (IFAAMAS), 2014, S. 277–284.
- [WH08] F. Wu und B.A. Huberman. „How public opinion forms“. In: 5385 LNCS (2008), S. 334–341.
- [YS17] K. Yang und J. Stoyanovich. „Measuring fairness in ranked outputs“. In: *ACM International Conference Proceeding Series*. Bd. Part F128636. ACM, 2017.
- [Zeh+17] M. Zehlike u. a. „FA IR: A fair top-k ranking algorithm“. In: *International Conference on Information and Knowledge Management, Proceedings*. Bd. Part F131841. ACM, 2017, S. 1569–1578.
- [Zha+11] D. Zhang u. a. „How to count thumb-ups and thumb-downs: User-rating based ranking of items from an axiomatic perspective“. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 6931 LNCS (2011), S. 238–249.

A. Modellkonfigurationen

Die Grundkonfiguration in Konfiguration A.1 definiert alle in der Simulation notwendigen Parameter. Diese können durch die Angabe einer weiteren Konfiguration überschrieben werden. Die Grundkonfiguration ist für die beiden unterschiedlichen Votingsystem $V_N = 1, 2$ gleich.

```
basic_config = Dict(
    :activity_distribution => Beta(2.5,5),
    :concentration_distribution => Poisson(50),
    :deviation_function => no_deviation,
    :gravity => 0,
    :init_score => 0,
    :new_posts_per_step => 10,
    :quality_distribution => Distributions.MvNormal(
        zeros(quality_dimensions),
        I(quality_dimensions),
    ),
    :relevance_gravity => 0,
    :rating_metric => metric_hacker_news,
    :start_posts => 100,
    :start_users => 100,
    :steps => 100,
    :user_opinion_function => consensus,
    :vote_evaluation => vote_difference,
    :voting_probability_distribution => Beta(2.5,5),
)

```

Konfiguration A.1.: Grundkonfiguration

A. Modellkonfigurationen

Die Konfiguration Überblick gibt einen Überblick über die Bewertungsmetriken und wie diese auf die Veränderung von Parametern reagieren. Durch diese Konfiguration werden 1152 Modellkonfigurationen definiert.

```
1    overview_config = [(
2        [upvote_system, up_and_downvote_system],
3        Dict(
4            :rating_metric => [
5                metric_view,
6                metric_hacker_news,
7                metric_activation,
8                metric_reddit_hot],
9                :init_score => [0, 70, 30000],
10            ),
11        ),
12        (
13            :all_models, Dict(
14                :deviation_function => [no_deviation, mean_deviation],
15                :vote_evaluation => [
16                    vote_difference,
17                    vote_partition,
18                    wilson_score],
19                :relevance_gravity => [0, 2],
20                :gravity => [0, 2],
21                :user_opinion_function => [
22                    consensus,
23                    dissent],
24            ),
25            ),
26        ],
27    ]
```

Konfiguration A.2.: Überblick

A. Modellkonfigurationen

Die Konfiguration Initialscore dient zur Veranschaulichung der unterschiedlichen Auswirkung der Variierung des Initialscores auf die Bewertungsmetriken. Es werden 26 Modellkonfigurationen definiert.

```
1 init_score_config = [(
2     :all_models,
3     Dict(
4         :relevance_gravity => 0,
5         :user_opinion_function => consensus,
6     )),
7     (
8         up_and_downvote_system,
9         Dict(
10            :rating_metric => metric_hacker_news,
11            :vote_evaluation => vote_difference,
12            :deviation_function => no_deviation,
13            :gravity => 0,
14            :init_score => [10, 20, 30, 40, 50, 70, 90, 110],
15        )),
16        (
17            up_and_downvote_system,
18            Dict(
19                :rating_metric => metric_activation,
20                :vote_evaluation => vote_difference,
21                :deviation_function => no_deviation,
22                :gravity => 2,
23                :init_score => [10, 20, 30, 40, 50, 70, 90, 110],
24            )),
25        (
26            up_and_downvote_system,
27            Dict(
28                :rating_metric => metric_view,
29                :vote_evaluation => vote_difference,
30                :deviation_function => mean_deviation,
31                :gravity => 0,
32                :init_score => [10, 20, 30, 40, 50, 70, 90, 110],
33            )),
34        (
35            up_and_downvote_system,
36            Dict(
37                :rating_metric => metric_reddit_hot,
38                :vote_evaluation => vote_difference,
39                :deviation_function => no_deviation,
40                :init_score => [0, 30000],
41            )),
42        )
43    )
44)]
```

Konfiguration A.3.: Initialscore

A. Modellkonfigurationen

Die Konfiguration Parametertest ist eine Zusammenstellung von jeder Bewertungsmetrik auf den Q&A- und News-Plattformen, auf welchen Konsens herrscht. Für jede Bewertungsmetrik wurden die Parameter so gewählt, dass die Bewertungsmetrik verhältnismäßig fair ist. Es sind 8 Modellkonfigurationen definiert. Um einzelne Parameter zu analysieren wird diese Konfiguration verwendet und der zu beobachtende Parameter, zum Beispiel die User*innenanzahl variiert.

```
1 parameter_test_config = [
2     (
3         up_and_downvote_system,
4         Dict(
5             :rating_metric => metric_hacker_news,
6             :init_score => 50,
7             :vote_evaluation => vote_difference,
8             :deviation_function => no_deviation,
9             :gravity => 0,
10            :relevance_gravity => 0,
11            :user_opinion_function => consensus,
12        ),
13    ),
14    (
15        up_and_downvote_system,
16        Dict(
17            :rating_metric => metric_hacker_news,
18            :init_score => 50,
19            :vote_evaluation => vote_difference,
20            :deviation_function => mean_deviation,
21            :gravity => 2,
22            :relevance_gravity => 2,
23            :user_opinion_function => consensus,
24        ),
25    ),
26    (
27        up_and_downvote_system,
28        Dict(
29            :rating_metric => metric_view,
30            :init_score => 20,
31            :vote_evaluation => vote_difference,
32            :deviation_function => mean_deviation,
33            :gravity => 0,
34            :relevance_gravity => 0,
35            :user_opinion_function => consensus,
36        ),
37    ),
38]
```

Konfiguration A.4.: Parametertest

```

1   (
2       up_and_downvote_system,
3       Dict(
4           :rating_metric => metric_view,
5           :init_score => 30,
6           :vote_evaluation => wilson_score,
7           :deviation_function => mean_deviation,
8           :gravity => 2,
9           :relevance_gravity => 2,
10          :user_opinion_function => consensus,
11      ),
12  ),
13  (
14      up_and_downvote_system,
15      Dict(
16          :rating_metric => metric_activation,
17          :init_score => 10,
18          :vote_evaluation => vote_difference,
19          :deviation_function => no_deviation,
20          :gravity => 2,
21          :relevance_gravity => 0,
22          :user_opinion_function => consensus,
23      ),
24  ),
25  (
26      up_and_downvote_system,
27      Dict(
28          :rating_metric => metric_activation,
29          :init_score => 30,
30          :vote_evaluation => vote_difference,
31          :deviation_function => mean_deviation,
32          :gravity => 2,
33          :relevance_gravity => 2,
34          :user_opinion_function => consensus,
35      ),
36  ),
37  (
38      up_and_downvote_system,
39      Dict(
40          :rating_metric => metric_reddit_hot,
41          :init_score => 30000,
42          :vote_evaluation => vote_difference,
43          :deviation_function => no_deviation,
44          :relevance_gravity => 0,
45          :user_opinion_function => consensus,
46      ),
47  ),

```

Konfiguration A.5.: Parametertest (2)

A. Modellkonfigurationen

```
2
3     up_and_downvote_system,
4     Dict(
5         :rating_metric => metric_reddit_hot,
6         :init_score => 30000,
7         :vote_evaluation => vote_difference,
8         :deviation_function => mean_deviation,
9         :relevance_gravity => 2,
10        :user_opinion_function => consensus,
11    ),
12 ]
```

Konfiguration A.6.: Parametertest (3)