

**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, 13. August 2020

---

Felix Matuschka

# Inhaltsverzeichnis

<b>Formelzeichen</b>	<b>i</b>
<b>1. Einleitung</b>	<b>2</b>
1.1. Motivation . . . . .	2
1.2. Agent*innenbasierte Modellierung . . . . .	3
<b>2. Verwandte Arbeiten</b>	<b>4</b>
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
<b>3. Modell von Votingsystemen</b>	<b>7</b>
3.1. Definition der Qualität . . . . .	7
3.2. Bewertungraum 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
3.5.1. Aktivität . . . . .	10
3.5.2. Konzentration . . . . .	10
3.5.3. Qualitätsperzeption . . . . .	11
3.5.4. Meinungsfunktion . . . . .	11
3.5.5. Meinungsverteilung . . . . .	11
3.5.6. Bewertungszufriedenheit . . . . .	11
3.6. User*innen und Postanzahl . . . . .	12
<b>4. User*innenmeinung</b>	<b>13</b>
4.1. Transformation der Qualitätsparameter . . . . .	13
4.2. Approxamtion der User*innenmeinungfunktion . . . . .	13
4.2.1. Meinung im Konsens . . . . .	14
4.2.2. Meinung im Dissens . . . . .	14
<b>5. Bewertungsmetriken</b>	<b>16</b>
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 . . . . .	17
5.3. Verallgemeinertes Hacker News . . . . .	18
5.4. Reddit Hot . . . . .	18
5.5. View Bewertungsmetrik . . . . .	19

## Inhaltsverzeichnis

5.6. Aktivität . . . . .	19
5.7. Zufallsbewertung . . . . .	19
5.7.1. $\mu$ -Abweichung . . . . .	19
5.7.2. $\sigma$ -Abweichung . . . . .	19
<b>6. Evaluationsmethode</b>	<b>20</b>
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
<b>7. Implementierung</b>	<b>22</b>
7.1. agent*innenbasierte Modellierung von Votingsystemen . . . . .	22
7.1.1. Agent*innenschrittfunktion . . . . .	23
7.1.2. Modellschrittfunktion . . . . .	23
7.2. Erstellung von Modellkonfigurationen . . . . .	24
7.3. Auswertung der Modelle . . . . .	25
<b>8. Simulationsergebnisse</b>	<b>27</b>
8.1. Größe des Bewertungsvektor . . . . .	28
8.2. Initialscore . . . . .	28
8.3. Gravität . . . . .	30
8.4. Bewertungstransformation . . . . .	30
8.5. Zufallsabweichung . . . . .	31
8.6. Modellparameter . . . . .	32
8.6.1. Iterationslänge . . . . .	32
8.6.2. Qualitätsraum . . . . .	32
8.6.3. User*innen- und Postanzahl . . . . .	33
8.6.4. Vorsortierung der Posts . . . . .	34
8.7. User*innenparameter . . . . .	34
<b>9. Diskussion der Ergebnisse</b>	<b>35</b>
<b>10. Fazit und Ausblick</b>	<b>38</b>
<b>Literatur</b>	<b>40</b>
<b>A. Modellkonfigurationen</b>	<b>42</b>

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

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 definiert und auf unterschiedlichen Arten von Plattformen, wie News- und Q&A-Plattformen, in einer agent\*innenbasierten Simulation eines sozialen Mediums 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

## Modellparameter

$M_N$	Iterationslänge
$i_M$	Iterationsschritt
$P$	Post
$U$	User*in
$R$	User*innenmeinungsfunktion
$W$	User*innenmeinungsverteilung
$V$	Bewertungsraum eines Votingsystems
$R_\gamma$	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
$\tau$	Transformation des Bewertungsvektors
$\gamma$	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

In den letzten Jahrzehnten 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. Dies führt zu einer Menge an Information, die kein Mensch überblicken oder verstehen kann. Es werden Algorithmen verwendet, welche die Beiträge filtern und den Nutzer\*innen diese anzeigen, welche sie wahrscheinlich interessant finden.

Um Informationen 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 Beitrag zu äußern. Hacker News<sup>1</sup> erlaubt User\*innen Posts nur positiv, mit Upvotes, zu bewerten. Auf Reddit<sup>2</sup> ist es ebenfalls möglich Posts mit Downvotes herabzuwerten. Onlineshops wie TheCubicle<sup>3</sup> verwenden zum Beispiel ein 1 bis 5 Sterne System um Artikel bewerten zu lassen.

Es können Empfehlungssysteme verwendet werden, diese zeigen Nutzer\*innen diese Beiträge personalisiert an, welche bereits andere Nutzer\*innen mit ähnlichen Interessen als gut bewertet haben. Während Nutzer\*innen aus ihrer Perspektive interessante Posts erhalten hat diese Methode einige negative Effekte. Die Anwendung führt zur Bildung von Filterblasen. User\*innen interagieren nur noch mit Beiträgen, welche sie positiv wahrnehmen. Nutzer\*innen welche 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, wenn allen Nutzer\*innen die gleichen Beiträge angezeigt werden. Die Beiträge müssen trotzdem anhand des Interesses für Nutzer\*innen gefiltert und sortiert werden. Dazu werden *Bewertungsmetriken* verwendet, welche die Posts anhand ihrer Interaktion, zum Beispiel unter Einbeziehung ihrer Upvotes und des Veröffentlichungszeitpunkt bewerten und anordnen.

Eine Problematik die auch bei Bewertungsmetriken auftreten ist der *Matthäus-Effekt*: Beiträge welche bereits viel Aufmerksamkeit erhalten haben, werden weiteren Nutzer\*innen angezeigt und erhalten dadurch weitere Aufmerksamkeit und Interaktion. Beiträge hingegen, welche wenig Aufmerksamkeit erhalten, werden durch die Bewertungsmetrik oder das Empfehlungssystem nicht weiter als interessant erachtet.

---

<sup>1</sup><https://news.ycombinator.com>

<sup>2</sup><https://www.reddit.com>

<sup>3</sup><https://www.thecubicle.com>

## *1. Einleitung*

Im Gegensatz zu Empfehlungssystemen basieren Bewertungsmetriken meist nicht auf Methoden des maschinellen Lernens, sondern auf einfachen Algorithmen, welche leicht austauschbar sind.

Es ist wünschenswert eine Bewertungsmetrik zu finden, welche die Matthäus-Effekte minimiert. Aufgrund ihrer einfachen Struktur können sie leicht simuliert und verglichen werden.

Es ist nicht ausgeschlossen, dass die Art des Votingssystems Einfluss auf die Matthäus-Effekte hat. Das Votingsystem sollte auf der Suche nach einer optimalen Bewertungsmetrik mit einbezogen werden.

In dieser Arbeit wird ein agent\*innen basiertes Modell von Votingsystemen in sozialen Medien entworfen um unterschiedliche Bewertungsmetriken zur 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 modelliert werden. So sind Agent\*innen nicht auf die Modellierung von Menschen beschränkt. Durch Agent\*innen können in einer Simulation eines Vogelschwams 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, welche 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 interagieren.

Die Popularität steigt stetig, agent\*innenbasierte Modelle sind beliebt um unter anderem biologische, ökonomische und soziale Systeme, wie soziale Medien zu modellieren. Auch in dieser Arbeit wird die agent\*inenbasierten 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 Rankings in sozialen Medien. Einige Erkenntnisse und Ideen dieser 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, welche andere Agent\*innen in ihrer Meinung beeinflussen und von anderen selbst beeinflusst werden. [MVN19] ist eine bibliographische Übersicht, welche die wichtigsten Charakteristiken solcher Modelle herausarbeitet.

Für eine Gruppe aus Individuen, welche alle eine eigene subjektive Meinung besitzen wird von [Deg74] ein Modell vorgeschlagen, welches die unterschiedlichen Meinungen zu einem Konsens zusammenführen kann.

In [TL14] wird ein agentenbasiertes Modell vorgestellt, welches skeptizistische Agent\*innen gegenüber anderen Meinungen beinhaltet. 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, 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 beiden Gruppen gleich ist.

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

Ein Algorithmus, welcher aus einer Menge von *beschützten* und *nicht beschützten* Objekten die qualitativsten  $k$  Objekte sucht wird in [Zeh+17]. 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 Rankings 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 Anwendung von unterschiedlichen Bewertungsmetriken Fairness amortisiert.

### 2.2.2. Individuelle Fairness in sozialen Medien

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 erhalten die Lieder nach der Downloadzahl sortiert mit angezeigter Downloadzahl zu sehen, während die Personen aus der unabhängigen Gruppe jeweils eine zufällige Liste ohne Information über die Downloadzahl angezeigt. Durch die Ergebnisse zeigen die Autor\*innen, dass unter sozialem Einfluss Ungleichheit und Unvorhersehbarkeit bezüglich der Popularität der Lieder besteht.

Muchnik, Aral und Taylor findet in [MAT13] heraus, dass sozialer Einfluss Bewertungsdy namiken verzerrt und zur Bildung von Bewertungsblasen führen kann. Negativer sozialer Einfluss kann durch die Gruppenintelligenz wieder ausgeglichen werden.

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

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

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

## 2. Verwandte Arbeiten

Für Votingsysteme mit den Bewertungsmöglichkeiten Up- und Downvotes eines Beitrages wird in [Mil] der *Wilson-Score* zur Verrechnung der Up- und Downvotes vorgeschlagen. Es wird gezeigt, weshalb andere Methoden schlechter geeignet sind. Der *Wilson-Score* findet im Reddit- "BestKommentarranking Anwendung. Zhang u. a. stellt intuitive Axiome vor, welche ein Verrechnung der Bewertungen erfüllen sollte. Bekannte Methoden wie der *Wilson-Score* werden auf die Axiome geprüft.

Die Autor\*innen von [HL12] und [LH10] stellen ein stochastisches Modell auf um die Popularität von Beiträgen auf der Plattform Digg<sup>1</sup> vorherzusagen. Modellparameter, wie die Aktivitätsverteilung von User\*innen wird durch einen Datensatz von Digg gefüllt. Das Modell wird validiert und erfasst die Hauptkomponenten der Bewertungsdynamiken von Digg

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

Kaltenbrunner, Gómez und López versucht in [KGL07] die Aktivität die ein Post auf der Plattform Slashdot<sup>2</sup> insgesamt erzeugt auf Grundlage der erzeugten Aktivität in den ersten Minuten/Stunden nach Veröffentlichung des Posts vorherzusagen.

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

In [DN11] werden Kommentarsysteme von Nachrichtenagenturen und 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 das Ranking von Hacker News einzufügen, um die scharfe Aufmerksamkeitskante zwischen Posts die auf der Startseite landen und denen auf den hinteren Seiten auszuglätteten.

Die Autoren von [CSV18] argumentieren, dass viele Bewertungsmetriken die Diversität verringern und Stereotypen reproduzieren, die Bewertungsmetriken jedoch nicht darauf geprüft werden. Es wird ein Algorithmus vorgeschlagen, welcher eine Optimierung zur Bestimmung einer Bewertungsmetrik unter bestimmten Fairnessbedingungen vornimmt.

---

<sup>1</sup><https://digg.com/>  
<sup>2</sup>

## 3. Modell von Votingsystemen

In diesem Modell von Votingsystemen können User\*innen Posts veröffentlichen. Diese sind für alle anderen User\*innen sichtbar und können bewertet werden. Die Posts werden nach einer Bewertungsmetrik unter Betrachtung der Postparameter, wie die Anzahl und Art der Bewertungen, Betrachtungen und Zeitpunkt der Veröffentlichung bewertet. Auf der Startseite der Plattform werden die Posts absteigend nach ihrer Bewertung sortiert in einer vertikalen Liste angezeigt. Je weiter ein\*e User\*in auf der Seite herunterscrollt, 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, welche User\*innen wahrnehmen. Die Postqualität besitzt mehrere Merkmale, wie zum Beispiel den Informationsgehalt, die Originalität oder die Richtigkeit der Grammatik und Rechtschreibung. User\*innen nehmen die bestimmte Qualitätsmerkmale unterschiedlich wahr. Manchen User\*innen erachten den Informationsgehalt eines Posts als wichtig, während andere eher auf die Witzigkeit 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 Qualitätsvektor beschreibt die Ausprägung der Qualität in einem Qualitätsmerkmal. Ein Qualitätsvektor der Dimension  $Q_N$  beschreibt kann somit ein mit  $Q_N$  Qualitätsmerkmalen beschreiben.

In einem Modell besitzen sämtliche Qualitätsvektoren die gleiche Dimension. Sie werden über eine kontinuierliche  $Q_N$ -dimensionale Verteilung  $Q$  mit dem Mittelwert  $\mu = 0$  erzeugt. Es bietet sich die Verwendung einer  $Q_N$ -dimensionalen Normalverteilung an. Einzelne Qualitätsmerkmale können so einfach in Korrelation gesetzt werden.

## 3.2. Bewertungraum 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, Im Reddit Hot Ranking 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  $V_N = \{1, 2\}$ .

## 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. Die beobachtbaren Parameter können in den Bewertungsmetriken verwendet werden. 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 des Postes 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, den der Post durch eine Bewertungsmetrik (Abschnit 5 zugeordnet wurde).

Der Initialscore  $\iota_0$  aller Posts 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 kann sich mit der Zeit ändern. Auf einer Newsplattform spielt Aktualität eine wichtige Rolle. Posts welche gerade erst veröffentlicht wurden besitzen meist eine höhere Relevanz als Posts, welche 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 meistens 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 Gravität gesetzte Alter des Posts. Durch die Gravität  $G_R$  wird festgelegt wie schnell die Posts an Relevanz verlieren:

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

Für eine Q&A-Seite ist  $G_R = 0$  anzunehmen, da so die Relevanz über die Zeit konstant bleibt, während für eine Newsseite  $G_R = 2$  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 stattdessen durch plausible Verteilungen definiert.

### 3.5.1. 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 online.

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

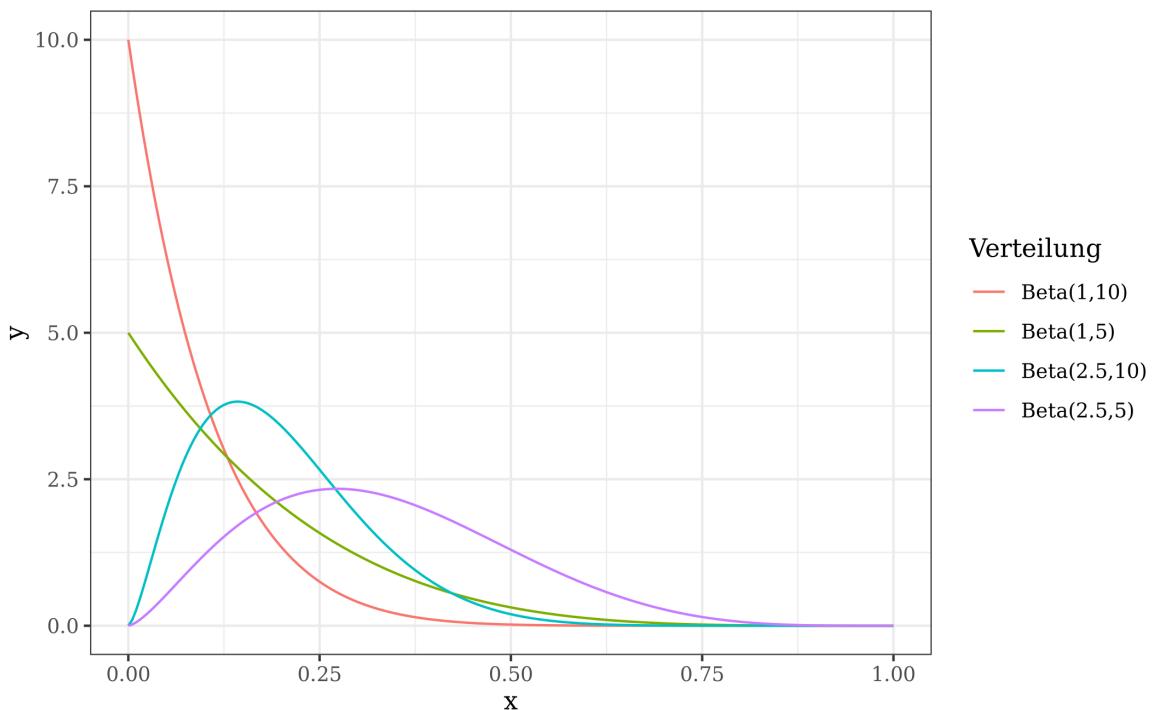


Abbildung 3.1.: Unterschiedliche  $\beta$ -Verteilungen zur Modellierung der User\*innenaktivität

### 3.5.2. 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 werden kann, um welchen die Konzentrationswerte abweichen.

### 3.5.3. 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 zu beurteilen.

$q_U$  wird als Zufallswert aus der Qualitätsverteilung erzeugt.

### 3.5.4. 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  $R(P, U) : \mathbb{R}^{Q_N} \rightarrow \mathbb{R}$  werden die beiden  $n_Q$ -dimensionalen Vektoren der Postqualität und Qualitätsperzeption auf einen skalaren Meinungswert  $m_{P,U} = R(P, U)$  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.

### 3.5.5. Meinungsverteilung

Die Verteilung der Meinungswerte  $W$  wird zur Hilfe gezogen, um zu entscheiden ob Posts von User\*innen bewertet werden. (Siehe dazu Abschnitt 3.5.6).

Zur Berechnung der empirischen Verteilungsfunktion 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  $1_{\{x \leq R(P_{S_i}, U_{S_j})\}} = 1$  für  $x \leq R(P_{S_i}, U_{S_j})$  und sonst 0.

### 3.5.6. 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 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  $Q2_W = 1 - \frac{z_U}{2}$ .

### 3. Modell von Votingsystemen

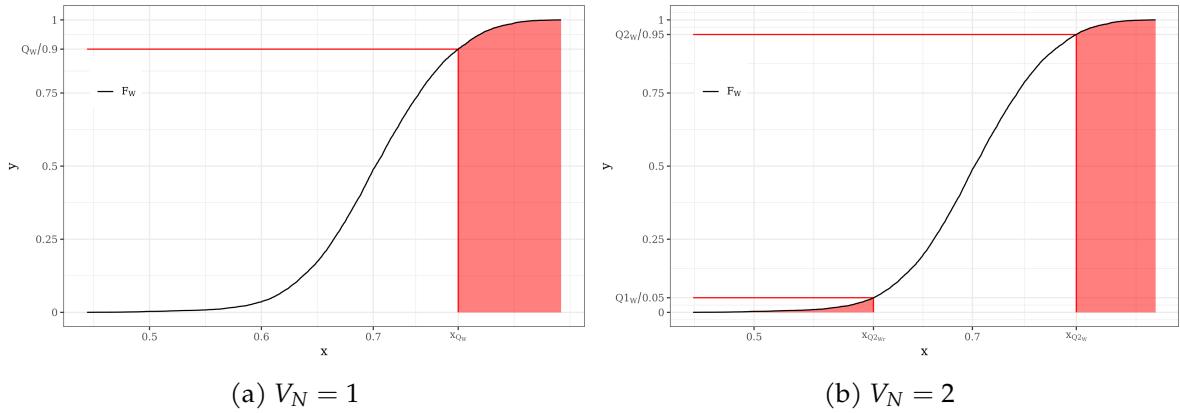


Abbildung 3.2.: Um eine Bewertung des Posts auszulösen muss der Meinungswert einer User\*in in den roten Bereich fallen

In Abbildung 3.2 sind beispielhafte Meinungverteilungen 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 für  $V_N = 2$  eingezeichnet 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 rotmarkierten 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  $\beta$ -Verteilungen, da so modelliert werden kann, dass viele User\*innen wenig Posts und wenige User\*innen viele Posts bewerten.

## 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 eine User\*in  $U$  die einen Post  $P$  betrachtet ist die Meinungsfunktion  $R(P, U) : \mathbb{R}^{N_Q} \rightarrow \mathbb{R}$ , welche die beiden  $N_Q$ -dimensionalen Vektoren der Postqualität und Qualitätsperzeption auf den skalaren Meinungswert  $m_{P,U} = R(P, U)$  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, welche 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 Post 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\*innenbewertungsfunktion ist hoch komplex und kann nur sehr vereinfacht modelliert werden. Im folgenden werden zwei Ansätze eingeführt um die User\*innenmeinungsfunktion  $R(P, U)$  zu approximieren.

#### 4. User\*innenmeinung

##### 4.2.1. Meinung im Konsens

In der Konsensbewertung wird davon ausgegangen, dass alle User\*innen "gute" Posts mit hohen Qualitätsmerkmalwerten eher als gut empfinden und Posts mit niedrigen Qualitätsmerkmalwerten eher als schlecht empfunden werden.

Die Konsensbewertung 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 die Konsensbewertung ausgedrückt:

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

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.

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

In Formel 4.5 wird die Dissensbewertung 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 für  $\tilde{q}_P = \tilde{q}_U$  einen Post maximal gut.

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

#### 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. 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  $U1$ .

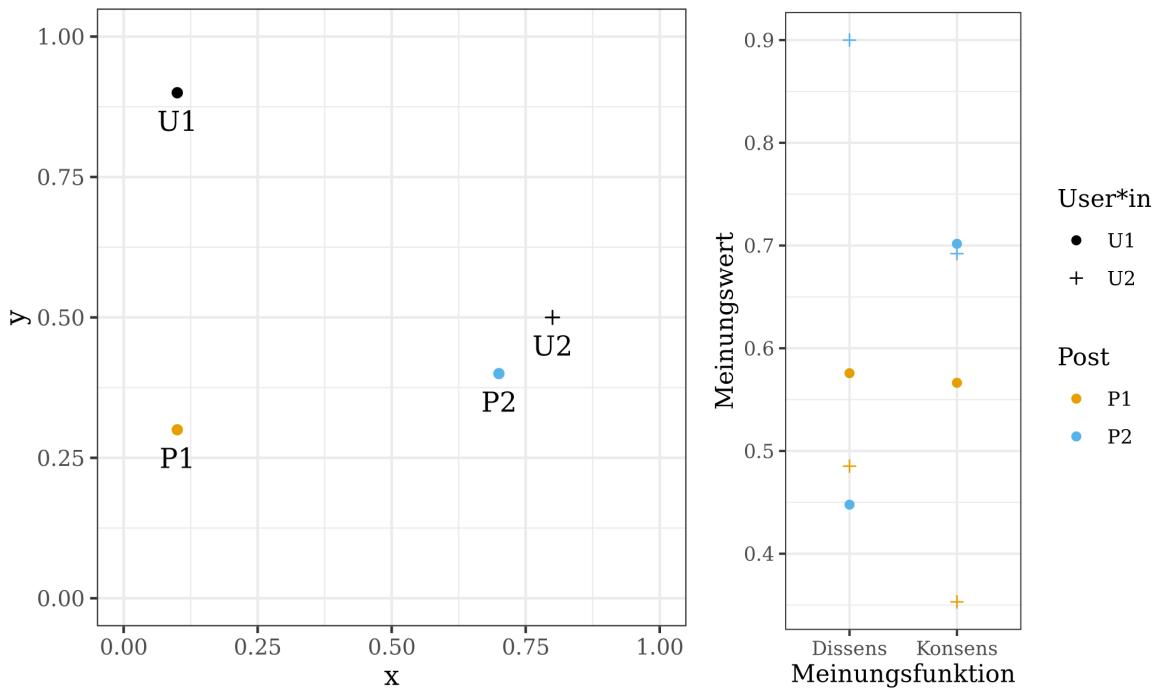


Abbildung 4.1.: Vergleich der beiden Meinungsfunktionen Konsens und Dissens

# 5. Bewertungsmetriken

Anhand von Bewertungsmetriken sollen Posts nach ihrer Relevanz sortiert werden. Ein Post  $P$  erhält durch eine Bewertungsmetrik den Score  $s_P = B(P)$  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 Betrachtungend

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

- für zwei Posts  $P_1$  und  $P_2$  mit  $R_{P_1} > R_{P_2}$  ist  $B(P_1) > B(P_2)$
- Für jeden Post  $P_i$  gilt  $v_{P_i} > 1$ , 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 im Reddit Hot Ranking 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 Metriken in bestimmten Konstellationen von Up- und Downvotes unintuitiv. Es wird die untere Grenze des Wilson-Konfidenzintervall für die Erfolgswahrscheinlichkeit der Binomialverteilung für den Parameter  $p$  als Metrik vorgeschlagen. Dabei ist  $n$  die Gesamtanzahl an abgegebenen Votes an einen Post, die Stichprobengröße, und  $k$  die Anzahl an positiver Bewertungen eines Postes, die Anzahl an Erfolgen in der Stichprobe.

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  $c_\alpha$  der Normalverteilung zum Irrtumsniveau  $\alpha$  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 Modellschritt  $i_M$ .

### 5.2.1. Hacker News

Hacker News verwendet eine Metrik, in der die Upvotes  $n_{\uparrow P}$  eines Postes ins Verhältnis zum Alter  $a_P = i_M - t_P$  setzt. Das Alter wird mit der Gravitationskonstante  $\gamma = 1.8$  potenziert. Die Bewertungstransformation lautet  $\tau(b_P) = u_P - 1$  so ergibt sich die Hacker News Bewertungs-metrik in Formel 5.4:

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

### 5.3. Verallgemeinertes Hacker News

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

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

### 5.4. Reddit Hot

Im Reddit Hot Ranking fließt die transformierte Bewertung  $\tau b_P$  eines Postes logarithmisch, und der Zeitpunkt der Veröffentlichung  $r_P$  einfach ein.

In der Reddit Hot Metrik wird ebenfalls die transformierte Bewertung  $\tau b_P$  eines Posts mit dem Alter verrechnet.

Das Alter  $e_P$  in der Reddit Hot Metrik 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 des Reddit Hot Ranking ist die Differenz aus Up- und Down-votes:

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

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 Bewertungsmetrik des Reddit Hot Rankings 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.

## 5.5. View Bewertungsmetrik

Die View Bewertungsmetrik basiert auf der in Kapitel 5.3 beschriebenen Metrik. Es fließt die Anzahl der Betrachtungen des Posts  $v$  mit ein. Daraus ergibt sich mit dem gleichen  $a_P$  wie in Abschnitt 5.2.1 die Bewertungsmetrik in 5.10:

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

## 5.6. Aktivität

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, so ergibt sich in 5.11:

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

## 5.7. Zufallsbewertung

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

### 5.7.1. $\mu$ -Abweichung

Sei  $\mu_s$  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:

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

### 5.7.2. $\sigma$ -Abweichung

Für die Standardabweichung der Scores aller Posts  $\sigma_s$  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 Userinnen\*parameter auswerten oder auch Funktionen auf Parametern ausführen. 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* in Formel 6.1 bestraft Bewertungsmetriken, welche Posts mit hoher Relevanz auf hintere Rankingpositionen einordnet. Die standardisierte Relevanz eines Posts  $\tilde{r}_P$  fließt durch den Logarithmus der Rankingposition des Posts ins Verhältnis gesetzt in den *DCG* ein.

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

**Normalized Discounted Cumulative Gain (nDCG)** Der *nDCG* normalisiert den *DCG* einer Bewertungsmetrik mit einer idealen Bewertungsmetrik  $B_I$ , welche die Posts absteigend nach Relevanz im Ranking anordnet.  $IDCG = DCG(B_I)$  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 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 andere Posts hingegen von keine\*r einzige\*n User\*in ist  $G \approx 1$  maximal. Der Gini-Koeffizient wird beschrieben durch:

$$G = \frac{1}{2S} \sum_{i=1}^{P_N} \sum_{j \neq i} |v_{P_i} - v_{P_j}| \quad (6.3)$$

## 6.2. Modellevaluation

Modellevaluationsfunktionen werden nach der vollständigen Simulation eines Modells ausgeführt.

**Modellparameter** Da sich Modellparameter im Laufe des Modells nicht ändern können, werden diese durch Modellevaluationsfunktionen ausgewertet.

### 6.2.1. Aggregation von Iterationsevaluationsfunktionen

Für ein Modell mit  $i_M$  Iterationsschritten können Modellevaluationsfunktionen über den von Iterationsevaluationsfunktionen erzeugten Datenvektor  $x$  aggregieren.

Die im vorherigen Abschnitt vorgestellten Iterationsevaluationen werden durch die Trapezregel in Formel 6.4 aggregiert und normiert:

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

### 6.2.2. Posts ohne Betrachtungen

Der Prozentsatz an Posts welche von keine\*r einzige\*n User\*in betrachtet wurden werden durch die Funktion in Formel 6.5 gezählt,  $P_N = P_{startN} + M_N P_{iterN}$  ist die Gesamtzahl der Posts.

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

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 dem Framework Agents.jl<sup>1</sup> für Julia<sup>2</sup>. Die verwendete Version von Agents.jl ist ein weitereintwickelter Fork der Verion 3.0.0.

Sämtliche Funktionalität ist in dem Paket VotingProtocols<sup>3</sup> 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.

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. 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 Itarationen 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
```

---

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

<sup>2</sup><https://julialang.org/>

<sup>3</sup><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 der Meinungswert  $m_{P,U}$  und bewertet wenn die in Abschnitt 3.5.6 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  $P_{Niter}$  neue Posts dem Modell hinzugefügt. definiert. 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 ein Framework entwickelt, welches es ermöglicht modular Modellparameter zu Modellkonfigurationen zusammenzustellen. Parameter können mehrdeutig überschrieben werden, für jeden mehrdeutigen Parameter wird eine eigene Modellkonfiguration erstellt und simuliert. Die Modellkonfigurationen werden durch eine Liste aus Tupeln definiert.

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

Listing 7.1.: Definition von Modellkonfigurationen in Julia

```

1 model_configs = [
2     (
3         downvote_model,
4         Dict(
5             :scoring_function => scoring_reddit_hot,
6             :deviation_function => [
7                 no_deviation,
8                 std_deviation],
9             ),
10        ),
11        (
12            [standard_model, downvote_model],
13            Dict(
14                :scoring_function => [
15                    scoring_activation,
16                    scoring_hacker_news],
17                :gravity => [0.5, 1.0, 1.5, 2.0],
18                ),
19            ),
20            (
21                :all_models,
22                Dict(
23                    :user_rating_function => user_rating_exp
24                    ),
25            ),
26        ]
27

```

Der erste Eintrag der Tupels definiert die Grundkonfiguration, diese stellt sicher, dass alle in der Simulation notwendigen Parameter gesetzt sind. Im zweiten Eintrag können über ein Dictionary einzelne Modellparameter überschrieben werden. In Zeile 3 wird das `downvote_model` ausgewählt. In den Zeilen 5 bis 7 werden einzelne Modellparameter überschrieben. Die Modellparameter werden als Symbole angesteuert. Der Modellparameter `deviation_function` wird mehrdeutig überschrieben, indem er durch eine Liste angegeben wird. Das erste Tupel definiert somit zwei Modellkonfigurationen, welche sich nur in `deviation_function` unterscheiden.

Der erste Eintrag des zweiten Tupels in Zeile 13 ist eine Liste von Grundkonfigurationen. Für jede der angegebenen Grundkonfigurationen werden die im zweiten Eintrag des Tupels definierten Modellparameter überschrieben. Die in Zeile 15 bis 18 angegeben Modellparameter

## 7. Implementierung

werden mehrdeutig überschrieben. `scoring_function` wird doppelt definiert und `gravity` vierfach. Dadurch werden pro Grundkonfiguration  $2 * 4 = 8$  Modellkonfigurationen erstellt. Da zwei Grundkonfigurationen angegeben sind werden durch das zweite Tupel insgesamt  $2 * (2 * 4) = 16$  Modellkonfigurationen festgelegt.

Das dritte Tupel 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_rating_function` mit `user_rating_exp` zugewiesen.

Insgesamt wurden durch `model_configs` 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 Arrays festgelegt. Die Evaluationsergebnisse werden analog in zwei Dataframes gespeichert und sind unter dem Namen der jeweiligen Evaluationsfunktion aufrufbar.

Modellevaluationsmethoden haben Zugriff auf das berechnete Datenstruktur 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.

Die Funktion `posts_with_no_views` berechnet  $P_{v=0}$  des Models. Das Macro `@area_under_aggregiert` die übergeben 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 R<sup>4</sup>. Die Datenanalyse kann so in R erfolgen.

---

<sup>4</sup><https://www.r-project.org/>

## 7. Implementierung

Listing 7.2.: Definition der Evaluationfunktionen

```
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(:scoring_function),
14    @model_parameter(:relevance_gravity),
15    @model_parameter(:user_rating_function),
16 ]
```

## 8. Simulationsergebnisse

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

Mit Blick auf die aufgestellten Kriterien einer fairen Bewertungsmetrik wird in der Auswertung der Koeffizient  $\rho$  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 Bewertungsmetriken minimiert und ist auf das Intervall  $[0, 1]$  beschränkt.

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

Mit den Modellkonfigurationen Überblick werden eine Q&A- ( $R_\gamma = 0$ ) und eine Newsplattform ( $R_\gamma = 2$ ) mit jeweils der User\*innenmeinungsfunktion Konsens  $R_K$ , für eine technische Plattform und Dissens  $R_D$ , für eine Diskussionsplattform, simuliert.

In Abbildung 8.1 sind die Simulationsergebnisse dargestellt. Jeder graue Punkt in der linken Abbildung beschreibt ein Modell, farblich hervorgehoben sind die Mittelwerte jeder einzelnen Modellkonfiguration aus Überblick. Die Farbe gibt Aufschluss über die verwendete Bewertungsmetrik. Auf der  $x$ -Achse ist der aggregierte Gini-Koeffizient  $T(G)$  angetragen, 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 Newsplattformen insbesondere mit der Dissens-Q&A-Plattform durch den Spearman-Koeffizienten von  $\rho$  korreliert sind.

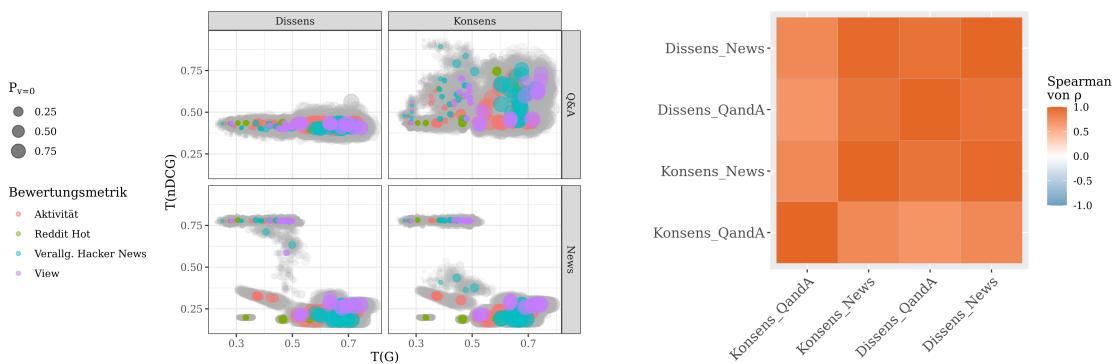


Abbildung 8.1.: Simulationsergebnisse nach vier Plattformarten. Die beiden News-Plattformen und die Q&A-Dissens-Plattform sind stark korreliert.

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 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  $\rho$  erzielt werden. Auf der News-Plattform nur in 65.6% der Konfigurationen. Das besten Ergebnisparameter werden jedoch mit  $V_N = 1$  auf der News-Plattform erzeugt.

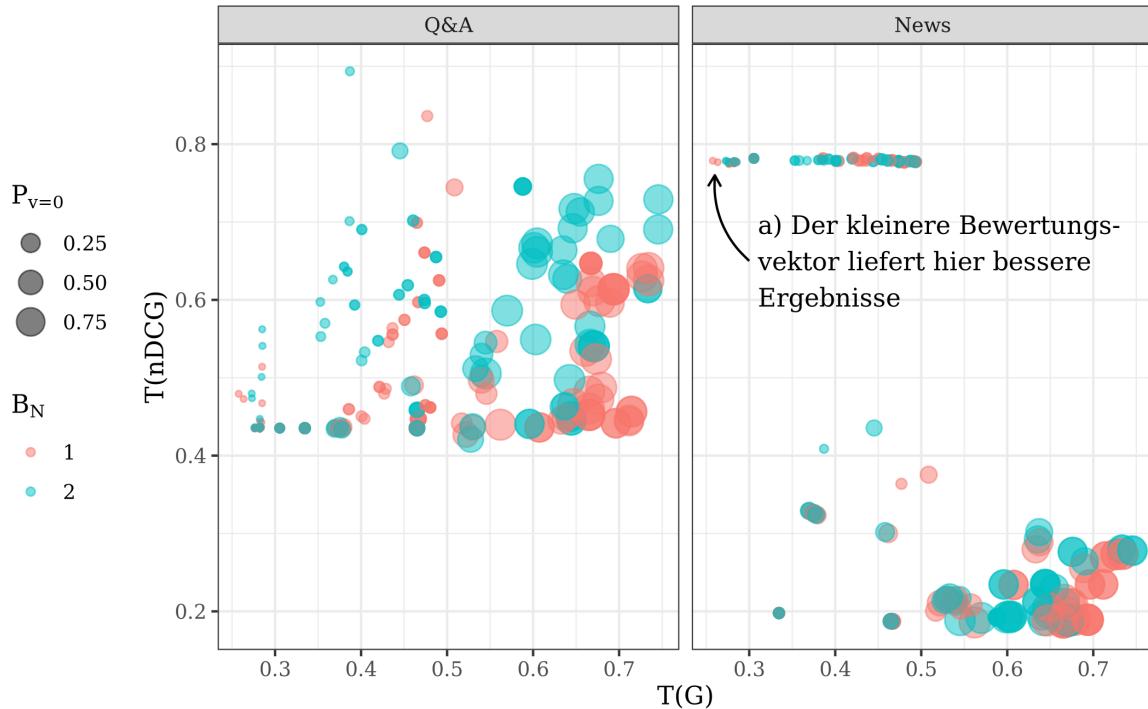


Abbildung 8.2.: In der Mehrheit der Konfigurationen ist der größere Bewertungsvektor überlegen.

## 8.2. Initialscore

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

Nach Konfiguration Initialscore ist der Einfluss auf die Bewertungsmetriken des Initialscores  $\iota_0$  farblich dargestellt. Die Veränderung von  $\iota_0$  wirkt sich unterschiedlich auf die vier Metriken aus.

In der Konfiguration des verallgemeinerten Hacker News Metrik in Abbildung wird durch die Erhöhung von  $\iota_0$   $T(G)$  und  $P_{v=0}$  verringert und  $T(nDCG)$  erhöht. Ab  $\iota_0 > 70$  wird  $T(nDCG)$  wieder verringert, während  $T(G)$  und  $P_{v=0}$  konstant bleiben. Bei der Aktivitätsmetrik führt eine Erhöhung von  $\iota_0$  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

## 8. Simulationsergebnisse

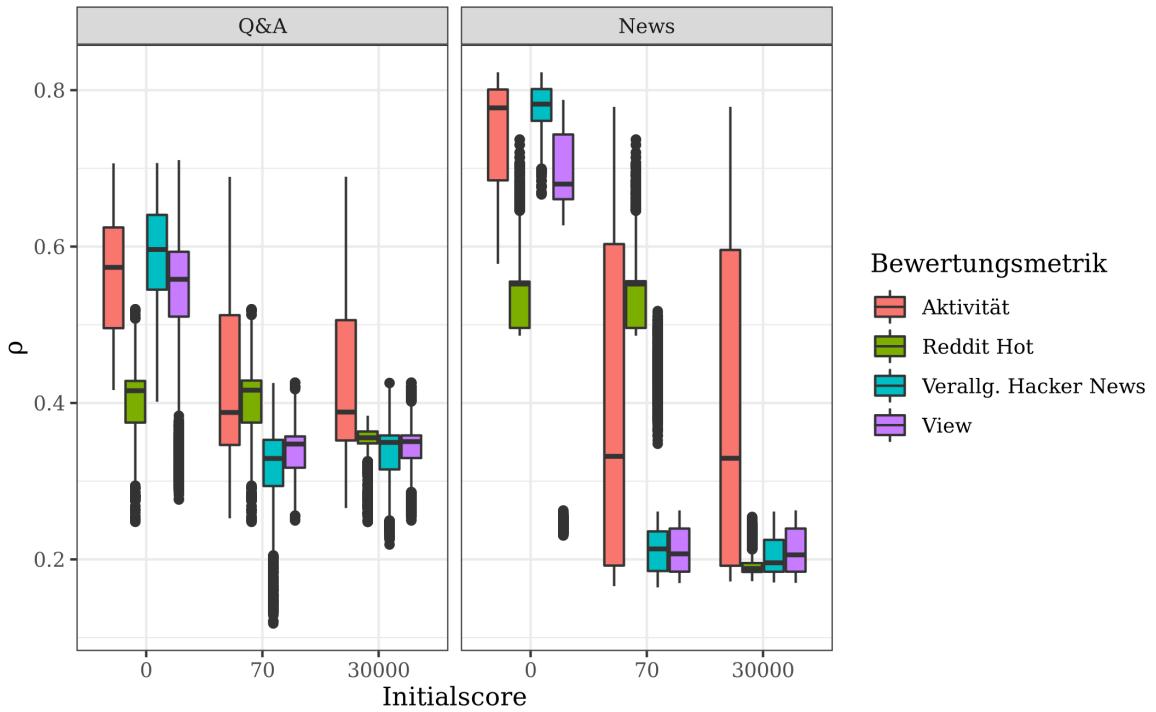


Abbildung 8.3.: Der Initialscore 0 ist für jede Metrik die schlechteste Wahl

anderen  $\iota_0$ . 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 Initialwert liefert bessere, kleinere  $T(G)$ ,  $T(nDCG)$  und  $P_{v=0}$ .

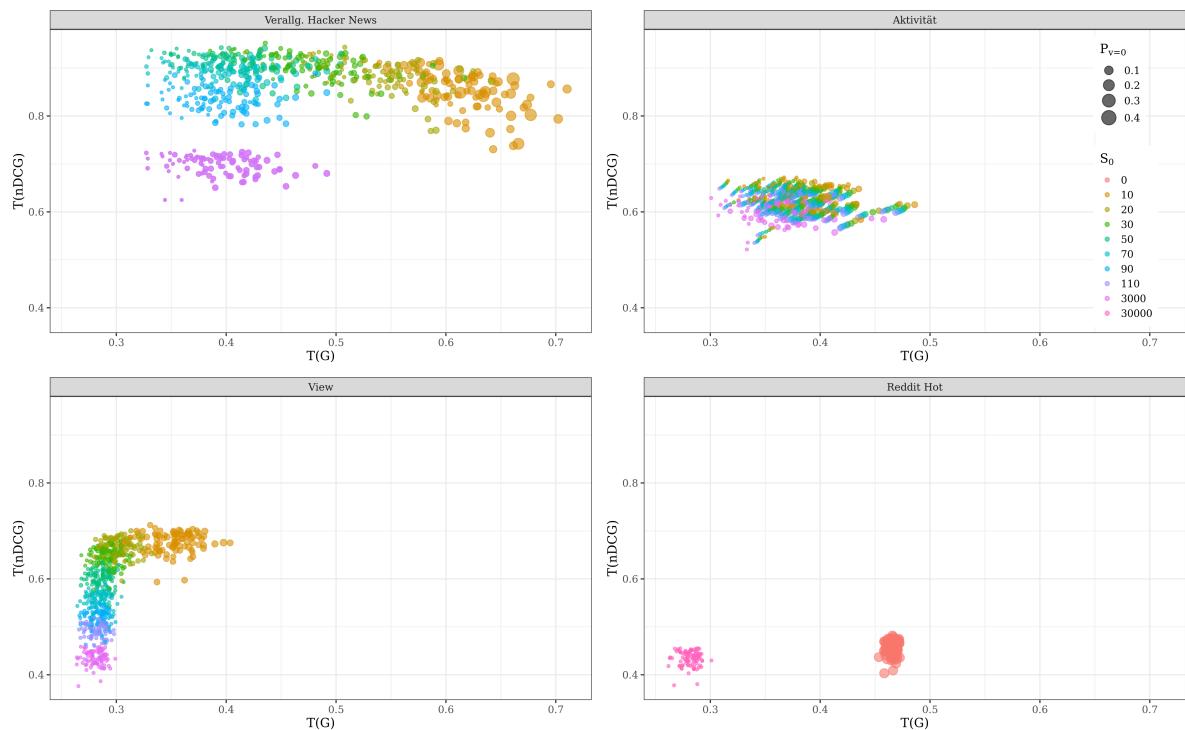


Abbildung 8.4.: Die Veränderung des Initialscores  $\iota_0$  wirkt sich unterschiedlich auf die vier Bewertungsmetriken aus.

### 8.3. Gravität

Der Einfluss der Gravität der drei Bewertungsmetriken mit Gravität  $G$  wird nach Konfiguration Parameteretest 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  $\rho$  wird bei  $\gamma = 0.5$  erzielt. Auf der Q&A-Plattform ist für die View- und Hacker News Metrik bei  $\gamma = 0$  der  $\rho$ -Wert am geringsten. Dieser steigt mit steigendem  $\gamma$  für alle drei Metriken an. Auf der News-Plattform hat die Gravität keinen signifikanten Einfluss auf die View Metrik. Für die beiden weiteren Metriken wird über  $\gamma \in [0, 2]$  der  $\rho$  – Wert signifikant verringert, ab  $\gamma > 2$  steigt  $\rho$  wieder geringfügig an.

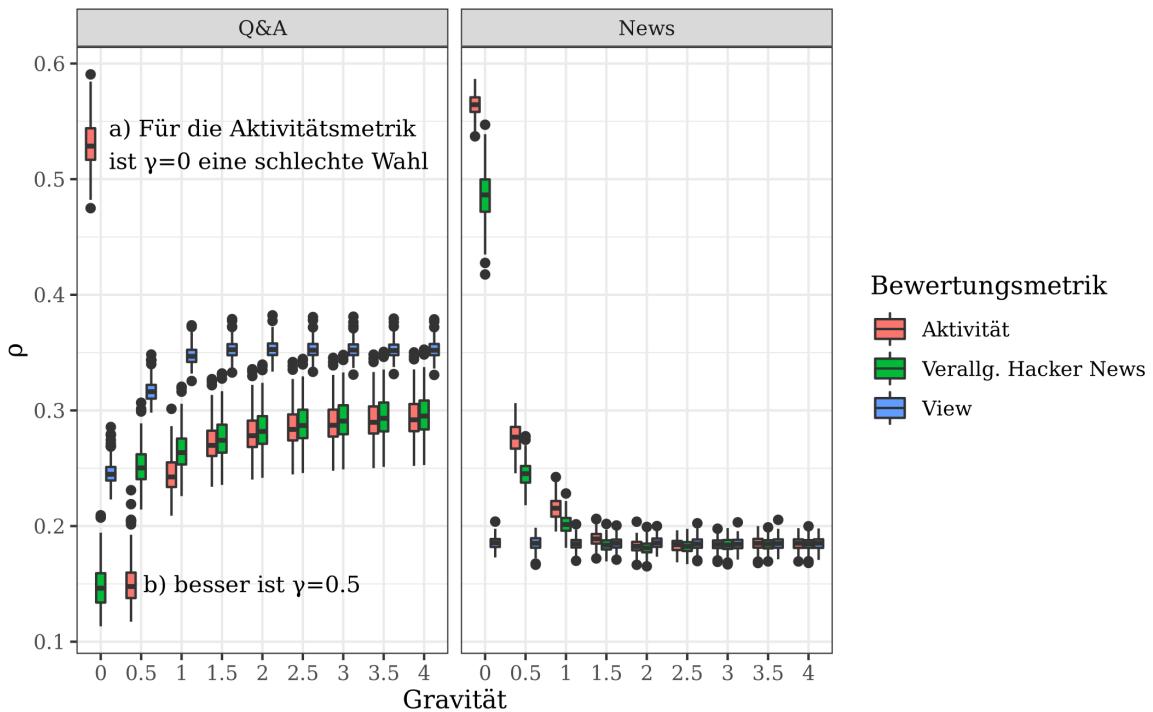


Abbildung 8.5.: Die Variierung der Gravität beeinflusst insbesondere auf die Aktivitäts- und Hacker News Metrik.

### 8.4. Bewertungstransformation

In Abbildung 8.6 ist die verwendete Bewertungstransformation in der Überblick-Konfiguration gekennzeichnet. Ein Datenpunkt beschreibt den Mittelwert einer Modellkonfiguration. Die Verallgemeinerte Hacker News Metrik ist mit  $VHN$ , die Reddit Hot Metrik mit  $RH$  abgekürzt. Es zeigt sich, dass  $\tau_{diff}$  in den meisten Bewertungsmetrikfällen eine größere Varianz bezüglich  $\rho$  als  $v_{anteil}$  und  $v_{wilson}$  aufweist. Auf der Q&A-Plattform besitzt  $\tau_{diff}$  aber stets einen geringeren Mittelwert. Auf der News-Plattform ist die Performance von  $\tau_{anteil}$  und  $\tau_{wilson}$  sehr ähnlich und meist besser als  $\tau_{diff}$ .

## 8. Simulationsergebnisse

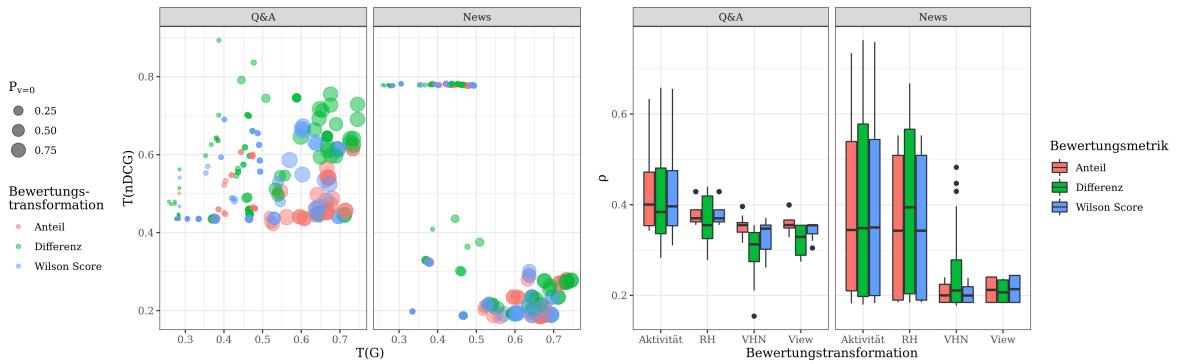


Abbildung 8.6.: Die Auswirkung der Bewertungstransformation. Auf einer Q&A-Plattform empfiehlt sich die Wahl von  $\tau_{diff}$

## 8.5. Zufallsabweichung

Die Boxplots in Abbildung 8.7 zeigen die Auswirkung der Zufallsabweichung in der Konfiguration Parameter test 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  $\sigma$ -Abweichung etwas mehr. Durch eine 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.

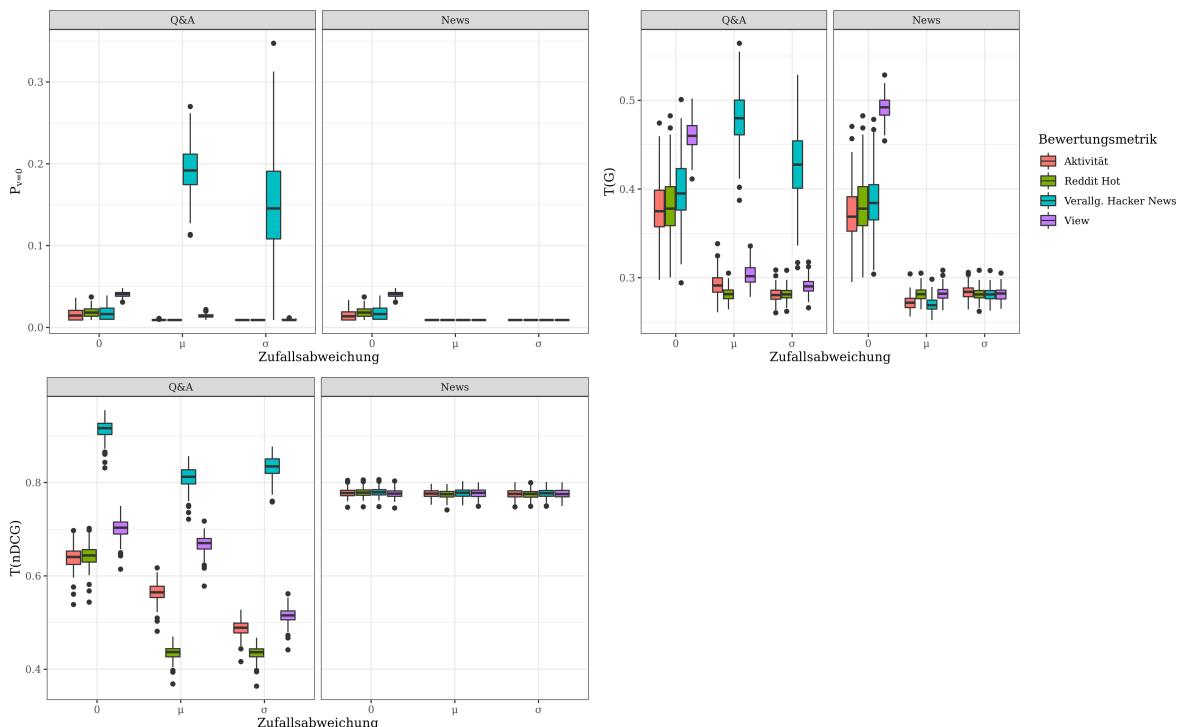


Abbildung 8.7.: Auf einer News-Plattform beeinflusst eine Zufallsabweichung die beobachteten Parameter positiv.

## 8.6. Modellparameter

### 8.6.1. Iterationslänge

In Abbildung 8.8 links oben sind die Simulationsergebnisse nach der Konfiguration Parametertest 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  $T(nDCGG), P_{v=0}$  und  $T(G)$ . 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. Die Varianz von  $T(nDCG)$  verringert sich für alle Metriken.  $P_{v=0}$  steigt für die Viewmetrik mit steigendem  $M_N$ . 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.

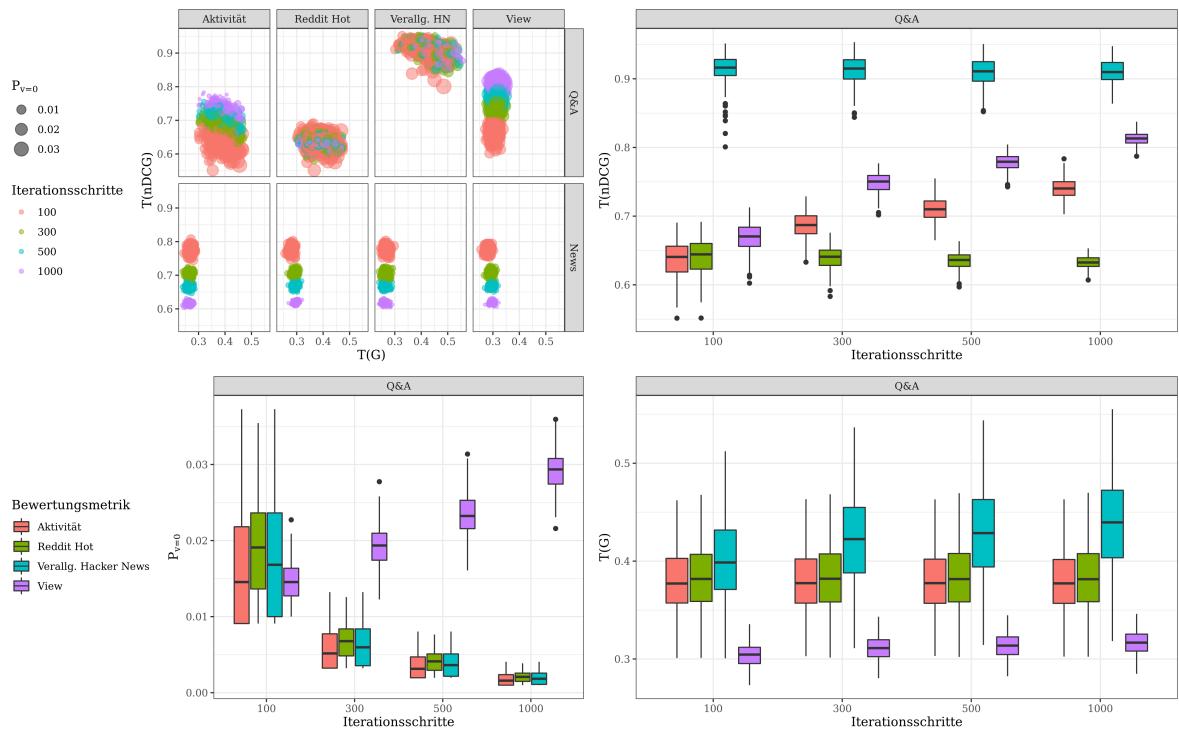


Abbildung 8.8.: Der Einfluss der Iterationslänge ist für die News-Plattform für alle Metriken gleich. Auf der Q&A-Plattform wirkt sich diese jedoch unterschiedlich aus.

### 8.6.2. Qualitätsraum

Der Einfluss der Größe des Qualitätsraum  $Q_N$  wird in Abbildung 8.9 nach den Ergebnissen der Konfiguration Parametertest 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  $\rho$  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  $N_Q$  bevorteilt.

## 8. Simulationsergebnisse

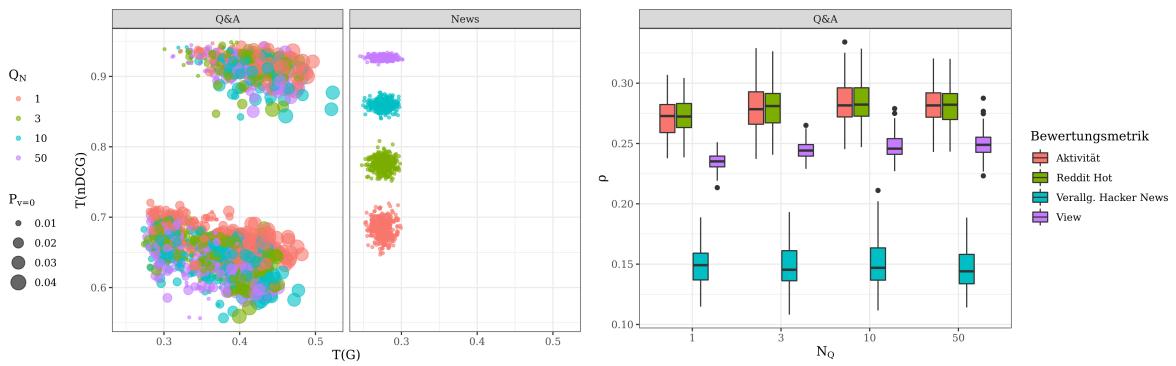


Abbildung 8.9.: Die Veränderung von  $Q_N$  wirkt sich nur auf die News-Plattform aus.

Wird die Qualitätsverteilung variiert, durch Änderung des Erwartungsvektor 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\*innen- und Postanzahl

In Abbildung 8.10 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 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  $\rho$  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.

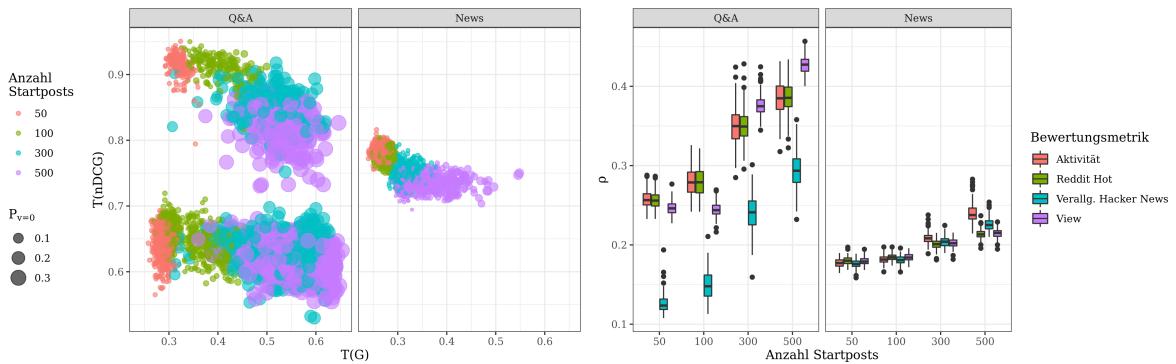


Abbildung 8.10.: Durch die Erhöhung der Anzahl von Startposts werden die Evaluationsparameter schlechter

#### 8.6.4. Vorsortierung der Posts

### 8.7. User\*innenparameter

Nach der *Parametertest*-Konfiguration sind Simulationsergebnisse in Abbildung 8.11a 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 8.11b sind die Dichteverteilungen der vier verwendeten Verteilungen dargestellt. Wie im besonderen an der Viewmetrik auf der Q&A-Plattform zu sehen ist, führt die Variierung 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, welche durch die Verwendung unterschiedlicher Verteilungen entstanden sind stehen in starker Korrelation.

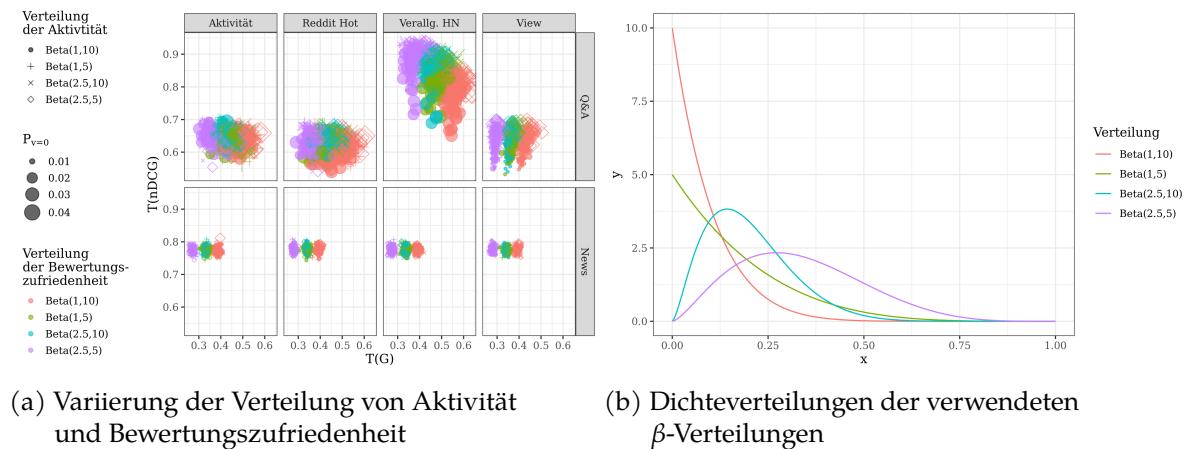


Abbildung 8.11.

Durch die Varriierung der Konzentration ergibt sich ein ähnliches Bild, wird der Erwartungswert der Konzentration erhöht, so verringern sich  $T(G)$  und  $P_{v=0}$ . Dies ist intuitiv, da User\*innen mehr Posts betrachten und so auch diese sehen, welche von der Bewertungsmetrik schlechter bewertet wurden.

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

## 9. Diskussion der Ergebnisse

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

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 vorgeschlagen um Up- und Downvotes eines Posts zu evaluieren, die Bewertung des Posts zu transformieren. In der Simulation zeigt sich, dass der Wilson Score nicht in allen Fällen der Differenz auf Up- und Downvotes überlegen. Auf einer Q&A-Plattform auf der Konsens herrscht ist die Differenz-Bewertungstransformation 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 Views. Sobald  $\iota_0 > 0$  gewählt wird, erhalten neue Posts von Beginn an Betrachtungen, gute Posts erhalten so die Chance gute Bewertungen zu erhalten. Für die Reddit Hot Metrik konnte kein optimaler Initialscore ermittelt werden, es ist jedoch zu vermuten das dieser, wie bei den drei weiteren Metriken, weder kleiner noch größer als die Scores 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  $\gamma = 0 + \epsilon, \epsilon = 0, 0.5$  vorteilhaft sind, ist auf einer News-Plattform 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. Auch der Gini-Koeffizient achtet implizit

## 9. Diskussion der Ergebnisse

auf die Anzahl der Posts ohne Betrachtungen, worauf diese hohe Korrelation zurückzuführen ist.

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 (technische Plattform) oder als Dissens (Diskussionsplattform) 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. Die User\*innenmeinungsfunktionen betrachtet nur die Qualität eines Postes und nicht das Alter, dieses wird jedoch bei einer hohen Relevanzgravität 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 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 sämtliche Modellkonfigurationen diese Verschiebung, sodass keine einzelnen Modellkonfigurationen durch die bestimmte Wahl der User\*innenparameter bevorteilt wird.

Die Erhöhung der Iterationsanzahl führt erwartungsgemäß zu einer Verringerung der Varianz und damit Präzisierung der betrachteten Ergebnisparameter. Im Fall von Q&A-Plattformen werden einzelne Bewertungsmetriken von hohen Iterationszahlen bevorzugt, sie liefern dort bessere Ergebnisse. Die Veränderung mit steigender Iterationszahl ist jedoch kontinuierlich, es treten keine unerwarteten Entwicklungen auf. Um die Langzeitperformance der Bewertungsmetriken zu analysieren ist es durchaus interessant die Iterationszahl weiter zu erhöhen.

Erfreulicherweise scheinen die Änderungen des Qualitätsraum keinen Einfluss auf die Vergleichbarkeit der Ergebnisse, 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 beliebig gewählt werden, eine realitätsgtreue Modellierung ist nicht notwendig.

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

## *9. Diskussion der Ergebnisse*

zu überprüfen ob die Vergleichbarkeit der Ergebnisse erhalten bleibt, sodass in der Simulation weiterhin kleine User\*innen- und Postanzahlen verwendet werden können.

## 10. Fazit und Ausblick

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 stochastisch modelliert. Es werden vier Bewertungsmetriken, darunter die Reddit Hot und die Hacker News Metrik eingeführt. Die Bewertungsmetrik mit dem Initialscore, der Zufallsabweichung und der Bewertungstransformation weitere variable Parameter. Für diese Bewertungsmetriken wurde ein Fairnessbegriff eingeführt, anhand dessen die Bewertungsmetriken vergleichbar werden. Über ein in Julia entwickeltes Framework können Modellkonfigurationen definiert und berechnet werden. Die Simulationsergebnisse zeigen, dass die Bewertungsmetriken sich in unterschiedlichen Konfigurationen in der Fairness unterscheiden. Die Auswahl der idealen Bewertungsmetrik hängt von der Art der Social Media Plattform ab.

In dieser Arbeit wurde eine agent\*innenbasiertes Modell von Votingsystemen zur Simulation einer Social Media Plattform entwickelt und in Julia implementiert. User\*innen können auf dieser Plattform mit Posts interagieren, indem sie diese auf durch das Votingsystem vorgebene Art bewerten. Mithilfe dieser Simulation können Bewertungsmetriken nach einem definiertem Fairnessbegriff verglichen und analysiert werden. Durch ein in Julia entwickeltes Framework können unterschiedliche Konfigurationen des Modells und Bewertungsmetriken definiert und simuliert werden. Ausgewählte Bewertungsmetriken, darunter die Reddit Hot und die Hacker News Metrik werden vorgestellt auf unterschiedlichen Plattformarten, Q&A- und News-Plattformen, jene auf welchen Konsens oder Dissens herrscht, berechnet. Die Ergebnisse zeigen auf, dass Bewertungsmetriken sich durchaus in der Fairness unterscheiden und die Wahl der idealen Bewertungsmetrik von der Art der Plattform abhängig ist.

# 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 fairesten ist.

Die Implementierung ist beschränkt auf Votingsysteme mit  $B_N = 1, 2$ , Modelle welche User\*innen Posts nur positiv oder negativ bewerten können. Es zeigt sich, dass die duale Bewertungsmodelle in mehr Konfigurationen bessere Ergebnisse liefert. Es ist interessant zu untersuchen wie sich Modelle mit größeren  $B_N > 2$  verhalten.

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

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 realem 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.

Die Framework bietet die einfache Definition von neuen Bewertungsmetriken und deren Parameter. In [Die15] wird das *Dirichlet Smoothing* verwendet und 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 Initialwert erhalten.

Der Featureraum für die Parameter der Bewertungsmetrik und des Votingsystems kann begrenzt werden, sodass ein Optimierungsproblem formuliert werden kann und zum Beispiel nach der Minimierung von  $\rho$  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).
- [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.
- [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.
- [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 Konfiguration Überblick gibt einen Überblick über die Bewertungsmetriken und wie diese auf die Veränderung von Parametern reagieren.

Konfiguration A.1.: Überblick

```
model_configs = [
    [standard_model, downvote_model],
    Dict(
        :scoring_function => [
            scoring_view,
            scoring_hacker_news,
            scoring_activation,
            scoring_reddit_hot],
        :init_score => [0, 70, 30000],
    ),
),
(
    :all_models, Dict(
        :deviation_function => [no_deviation, mean_deviation],
        :vote_evaluation => [
            vote_difference,
            vote_partition,
            wilson_score],
        :relevance_gravity => [0, 2],
        :gravity => [0, 2],
        :user_rating_function => [
            user_rating_exp2,
            user_rating_dist2],
    ),
),
]
]
```

## A. Modellkonfigurationen

Die Konfiguration Initialscore dient zur Verauschaulichung der unterschiedlichen Auswirkung der Variierung des Initialscores auf die Bewertungsmetriken

Konfiguration A.2.: Initialscore

```
1 model_configs = [(
2     :all_models,
3     Dict(
4         :relevance_gravity => 0,
5         :user_rating_function => user_rating_exp2,
6     )),
7     (
8         downvote_model,
9         Dict(
10            :scoring_function => scoring_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            downvote_model,
18            Dict(
19                :scoring_function => scoring_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            downvote_model,
27            Dict(
28                :scoring_function => scoring_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            downvote_model,
36            Dict(
37                :scoring_function => scoring_reddit_hot,
38                :vote_evaluation => vote_difference,
39                :deviation_function => no_deviation,
40                :init_score => [0, 30000],
41            ),
42        ),
43    )
44 ]
45 )]
```

## 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.

Konfiguration A.3.: Parametertest

```
1 default_models = [
2     (
3         downvote_model,
4         Dict(
5             :scoring_function => scoring_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_rating_function => user_rating_exp2,
12        ),
13    ),
14    (
15        downvote_model,
16        Dict(
17            :scoring_function => scoring_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_rating_function => user_rating_exp2,
24        ),
25    ),
26    (
27        downvote_model,
28        Dict(
29            :scoring_function => scoring_view,
30            :init_score => 20,
31            :vote_evaluation => vote_difference,
32            :deviation_function => mean_deviation,
33            :gravity => 0,
34            :relevance_gravity => 0,
35            :user_rating_function => user_rating_exp2,
36        ),
37    ),
38 ]
```

Konfiguration A.4.: Parametertest

```

1 default_models = [
2     (
3         downvote_model,
4         Dict(
5             :scoring_function => scoring_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_rating_function => user_rating_exp2,
12        ),
13    ),
14    (
15        downvote_model,
16        Dict(
17            :scoring_function => scoring_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_rating_function => user_rating_exp2,
24        ),
25    ),
26    (
27        downvote_model,
28        Dict(
29            :scoring_function => scoring_view,
30            :init_score => 20,
31            :vote_evaluation => vote_difference,
32            :deviation_function => mean_deviation,
33            :gravity => 0,
34            :relevance_gravity => 0,
35            :user_rating_function => user_rating_exp2,
36        ),
37    ),
38    (
39        downvote_model,
40        Dict(
41            :scoring_function => scoring_view,
42            :init_score => 30,
43            :vote_evaluation => wilson_score,
44            :deviation_function => mean_deviation,
45            :gravity => 2,
46            :relevance_gravity => 2,
47            :user_rating_function => user_rating_exp2,

```

Konfiguration A.5.: Parametertest

```

1      ),
2      (
3          downvote_model,
4          Dict(
5              :scoring_function => scoring_activation,
6              :init_score => 10,
7              :vote_evaluation => vote_difference,
8              :deviation_function => no_deviation,
9              :gravity => 2,
10             :relevance_gravity => 0,
11             :user_rating_function => user_rating_exp2,
12         ),
13     ),
14     (
15         downvote_model,
16         Dict(
17             :scoring_function => scoring_activation,
18             :init_score => 30,
19             :vote_evaluation => vote_difference,
20             :deviation_function => mean_deviation,
21             :gravity => 2,
22             :relevance_gravity => 2,
23             :user_rating_function => user_rating_exp2,
24         ),
25     ),
26     (
27         downvote_model,
28         Dict(
29             :scoring_function => scoring_reddit_hot,
30             :init_score => 30000,
31             :vote_evaluation => vote_difference,
32             :deviation_function => no_deviation,
33             :relevance_gravity => 0,
34             :user_rating_function => user_rating_exp2,
35         ),
36     ),
37     (
38         downvote_model,
39         Dict(
40             :scoring_function => scoring_reddit_hot,
41             :init_score => 30000,
42             :vote_evaluation => vote_difference,
43             :deviation_function => mean_deviation,
44             :relevance_gravity => 2,
45             :user_rating_function => user_rating_exp2,
46         ),
47     )
48 ]
49

```