

Zusammenfassung Die Anzahl an Beiträgen auf sozialen Medien ist für keinen Menschen überschaubar. Um ein gutes Benutzungserlebnis zu gewährleisten werden die Posts vorsortiert, dabei sollen gute und interessante Posts vielen User*innen angezeigt werden. Für die Sortierung werden Bewertungsmetriken verwendet, welche die Posts anhand ihrer Eigenschaften, zum Beispiel der Anzahl und Art der Bewertungen, sortieren. In dieser Arbeit wird ein Framework zur agent*innenbasiertes Modellierung einer Social Media Plattform entwickelt. Dabei werden sowohl Diskussionsplattformen als auch Wikis und Newsseiten betrachtet. Auf diesen wird das unterschiedlicher Bewertungsmetriken simuliert und verglichen. Dazu wird die Fairness einer Bewertungsmetrik definiert und schließlich ausgewertet. Es zeigt sich, dass die untersuchten Bewertungsmetriken unterschiedlich fair sind und die Wahl der idealen Bewertungsmetrik abhängig von der Art der Plattform ist.

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.

Viele soziale Medien und Onlineplattformen verwenden *Votingsysteme* um Nutzer*innen die Möglichkeit zu geben ihrer Meinung über den Beitrag zu äußern. Hacker News¹ erlaubt User*innen Posts nur positiv, mit Upvotes, zu bewerten. Auf Reddit² ist es ebenfalls möglich Posts mit Downvotes runterzuwerten. Onlineshops wie TheCubicle³ verwenden ein 1 bis 5 Sterne System um Artikel zu bewerten.

Es können Empfehlungssysteme verwendet werden, welche Nutzer*innen die Beiträge personalisiert anzeigen, welche bereits andere Nutzer*innen mit ähnlichen Interessen als gut bewertet haben. Während User*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.

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 das Empfehlungssystem nicht weiter als interessant erachtet. Diese Effekt wird *Matthäus-Effekt* bezeichnet.

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.

¹<https://news.ycombinator.com>

²<https://www.reddit.com>

³<https://www.thecubicle.com>

Auch Bewertungsmetriken sind Matthäus-Effekten ausgesetzt. Im Gegensatz zu Empfehlungssystemen basieren diese 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, den 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 Vogelschwarmes die Vögel modelliert werden. Für die Modellierung eines Waldbrandes würden die Agent*innen Bäume repräsentieren. Einzelne Agent*innen können sich 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.

Agent*innenbasierte Modelle sind 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 scheinbar 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*innenbasierten Modellierung zur Simulation einer Social Media Plattform verwendet.

1.3 Die modellierte Kommunikationsplattform

User*innen können 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.

2 Verwandte Arbeiten

Soziale Medien erfreuen sich immer größer User*innenzahlen, die Anzahl der Nutzung steigt und damit auch die Anzahl der geposteten Beiträge. Die Beiträge müssen, durch Bewertungsmetriken vorsortiert, den User*innen angezeigt werden um eine Grundqualität der Seiten zu wahren. Ausgewählte Arbeiten, die sich mit der Fairness von Bewertungsmetriken beschäftigen werden vorgestellt. Agentenbasierte Modellierung findet viel in der Modellierung von Meinungsdynamiken Anwendung. Interessante Arbeiten aus diesem Bereich werden ebenfalls vorgestellt.

2.1 Meinungsdynamiken

Um Meingungsdynamiken und Meinungsbildungsprozesse zu modellieren werden häufig agentenbasierte Modelle verwendet. Dabei werden Agent*innen als meinungshabende Individuen modelliert, welche andere Agent*innen in ihrer Meinung beeinflussen und von anderen selbst beeinflusst werden. [Agent-Based Models for Opinion Formation: A Bibliographic Survey] ist eine Bibliographische Übersicht, welche die wichtigsten Charakteristiken solcher Modelle herausarbeitet.

Die Autor*innen von [Mixing beliefs among interacting agents] untersuchen das Verhalten von Individuen, welche eine Meinung aus einem kontinuierlichem Raum besitzen. Es wird ein Modell vorgestellt um Meinungsbildung zu simulieren.

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

In [Opinion Dynamics of Skeptical Agents] wird ein agentenbasiertes Modell vorgestellt, welcken 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 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 [Fairness and Transparency in Ranking (nicht in Scopus)] stellen die Autor*innen fest, dass Fairness zwischen Gruppen hergestellt ist, wenn der Quotient aus Sichtbarkeit und Relevanz von Elementen der beiden Gruppen gleich ist.

[Fairness of Exposure in Rankings] und [Measuring Fairness in Ranked Output] betrachten Fairness in Rankings unter Einbeziehung von *beschützten* und *nicht beschützten* Objekten. Es werden Messfunktionen eingeführt für Fairness eingeführt um die Benachteiligung von *beschützten* Objekten systematisch auszuwerten.

Ein Algorithmus, welcher aus einer Menge von *beschützten* und *nicht beschützten* Objekten die relevantesten k Objekte sucht wird in [FA*IR: A Fair Top-k Ranking Algorithm vorgestellt] Für diese k Objekte sind *nicht beschützten* Objekte nicht benachteiligt.

Gruppenbezogene Fairness wird in [Equity of Attention: Amortizing Individual Fairness in Rankings] 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 Bewertungsmeirik nicht optimieren lässt. Sie stellen ein Optimierungsproblem auf, welche durch die Anordnung von unterschiedlichen Bewertungsmeiriken Fairness *amorisiert*.

2.2.2 Individuelle Fairness in sozialen Medien

In Salganik2006854 und [Measuring and Optimizing Cultural Markets] 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*. die *unabhängige* Gruppe erhält keine weiteren Informationen, alle User kriegen eine zufällige Liste der Lieder angezeigt und können diese downloaden. Die Gruppe unter *sozialem Einfluss* wird zusätzlich in acht Welten unterteilt, jeder User kriegt eine Liste mit Liedern nach der Downloadzahl sortiert angezeigt. Die Downloadzahlen werden angezeigt. Die Autoren zeigen, dass unter sozialem Einfluss Ungleichheit und Unvorhersehbarkeit der Popularität von Objekten besteht.

In [Leveraging Position Bias to Improve Peer Recommendation] wurde ein Experiment durchgeführt, bei dem Proband*innen Posts anderen Proband*innen empfehlen können, welche schließlich nach 5 unterschiedlichen Bewertungsmeiriken gerankt werden:

1. Zufall: in zufälliger Reihenfolge
2. Popularität: nach der Anzahl der Empfehlungen sortiert
3. Aktivität: nach der Zeit der letzten Empfehlung sortiert
4. Fixierung: stets in gleicher Reihenfolge
5. Fixierung invertiert: 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 ab, die fixierten Metriken am schlechtesten

In [How Not To Sort by Average Rating] wird der *Wilson-Score* zur Auswertung von Up- und Downvotes vorgeschlagen. Es wird gezeigt, weshalb andere Methoden schlechter geeignet sind. Der *Wilson-Score* findet im Reddit- "Best"-Kommentarranking Anwendung.

Für Rankings mit den Bewertungsmöglichkeiten "Daumen hoch" und "Daumen runter" stellt [How to Count Thumb-Ups and Thumb-Downs:User-Rating based Ranking of Itemsfrom an Axiomatic Perspective] intuitive Axiome vor, welche ein Auswertung der "Daumen" erfüllen sollte. Bekannte Methoden wie der *Wilson-Score* werden auf die Axiome geprüft.

Die Autor*innen stellen in [Social dynamics of Digg] und [Using an model of social dynamics to predict popularity of news] ein stochastisches Modell auf um die Popularität von Beiträgen auf der Platform *Digg* 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*

[Stoddard] 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. Sie entwickeln eine Methode um die Qualität von Posts auf den Plattformen abzuschätzen.

In [Description and Prediction of Slashdot Activity] wird versucht die Aktivität (Kommentierung) die ein Post auf der Platform *Slashdot* erzeugt auf Grundlage der erzeugten Aktivität in den ersten Minuten/Stunden nach Veröffentlichung des Posts.

Muchnik findet in **Muchnik2013647** heraus, dass sozialer Einfluss Bewertungsdynamiken verzerrt und zur Bildung von Bewertungsblasen führen kann. Negativer sozialer Einfluss kann durch die Gruppenintelligenz wieder ausgeglichen werden.

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

In [Towards Quality Discourse in Online News Comments] werden Kommentarsystem von Nachrichtenagenturen untersucht und die Wirkung von Beiträgen mit niedriger Qualität auf Nutzer*innen und Journalist*innen. 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 [Randomize HN] vor etwas Random Noise in das Ranking von Hacker News einzufügen, um die scharfe Aufmerksamkeitskante zwischen Posts die auf der Startseite landen und diesen, die auf den auszuglätteten.

2 Verwandte Arbeiten

Die Autoren von [Ranking with Fairness Constraints] argumentieren, dass viele Bewertungsmetriken die Diverität verringern und Stereotypen reproduzieren, die Bewertungsmetriken jedoch nicht darauf geprüft werden. Es wird ein Algorithmus vorgeschlagen, welcher eine Optimierung zur Bestimmung eines Objektrankings unter bestimmten Fairnessbedingungen vornimmt.

Diese Arbeit greift Ideen und Kritik zu bestehenden Bewertungsmetriken und deren Evaluation auf. Es wird ein Framework entwickelt, durch welches Bewertungsmetriken durch ein agentenbasiertes Modell simuliert und verglichen werden können.

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 Model (M) wird für eine festgelegte Anzahl Iterationsschritte N_M 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 n_Q modelliert. Jeder Eintrag eines Qualitätsvektor beschreibt die Ausprägung der Qualität in einem Qualitätsmerkmal. Ein Qualitätsvektor der Dimension n_Q beschreibt kann somit ein mit n_Q Qualitätsmerkmalen beschreiben.

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

3.2 Bewertungraum des Votingsystems

Ein Votingsystem kann einen Bewertungsraum B mit beliebiger Dimension n_B 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 $n_B = 2$. Viele Onlineshops lassen in einem fünfdimensionalem Raum Artikel mit 1 bis 5 Sternen bewerten.

Diese Arbeit ist beschränkt auf $n_B = \{1, 2\}$.

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 Postparameter

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

Bewertungsvektor Der Bewertungsvektor b_P eines Posts besitzt die Dimension n_B . In ihm wird Bewertung der User*innen des Postes gespeichert.

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

Der Initialscore s_0 aller Posts kann variiert werden.

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

3.3.2 Nicht beobachtbare Postparameter

Qualität Der n_Q -dimensionale Qualitätsvektor q_P beschreibt die Qualität eines Posts. Er wird aus der Qualitätsverteilung V_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 ein Newsplattform spielt Akualität eine wichtige Rolle. Posts welche gerade erst veröffentlicht wurden besitzen meist eine höhere Relevanz als Posts, welche vor langer Zeit veröffentlicht wurden. Auf eine Q&A-Seite verlieren Beiträge mit versteichender 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 3.1 ergibt sich somit durch die Summe aller Einträge vo 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:

$$R_P = \frac{1}{(i_M - a_P)^{G_R}} \sum_{i=1}^{N_Q} q_{P_i} \quad (3.1)$$

Für eine Q&A-Seite ist $G_R = 0$ anzunehmen, 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 Aktitvität a_P 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 β -verteilt. Einige denkbare β -Verteilungen sind in Abbildung ?? zu sehen. Da sie auf das Intervall $[0, 1]$ beschränkt sind β -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.

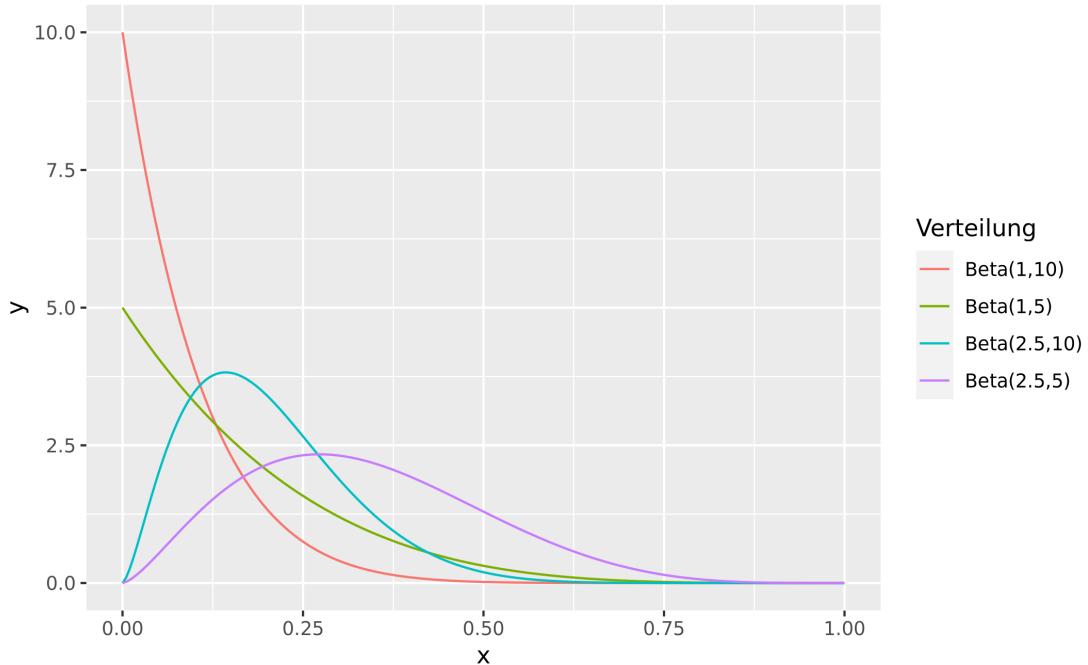


Figure 3.1: Unterschiedliche β -Verteilungen zur Modellierung der User*innenaktivität

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_P 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 n_Q 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 $R(P, U) : \mathbb{R}^{n_Q} \rightarrow \mathbb{R}$ 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 3.2 ergibt sich die empirische Verteilungsfunktion der verteilten Variable x :

$$F_W(x) = \frac{1}{n^2} \sum_{i,j=1}^n 1_{\{x \leq m_{P_j, U_i}\}} \quad (3.2)$$

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 β_U einer*r User*in definiert und hängt von der Dimension N_B des Bewertungsraumes ab. Für einen eindimensionalen Bewertungsraum ist $Q_W = 1 - \beta_U$.

In Abbildung ?? ist eine beispielhafte Meinungsverteilungsfunktion für $N_B = 1$ in schwarz dargestellt. Für β_U ist Q_W auf der y -Achse rot aufgetragen. Das Q_W -Quantil x_{Q_W} ist ebenfalls in rot auf der x -Achse eingezeichnet. Um eine Postbewertung herzurufen muss der Meinungswert des Posts größer als x_{Q_W} 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

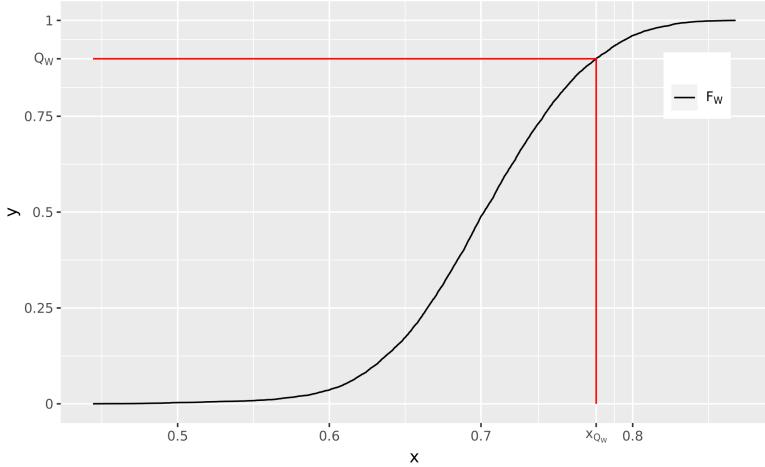


Figure 3.2: Um eine Bewertung einer User*in hervorzurufen, muss der Meingungswert $m_{P,U}$ größer als x_{Q_W} sein.

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:

N_U	Anzahl der User*innen
N_{Pstart}	Anzahl der Posts zu Beginn der Simulation
N_{Piter}	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.

4.2.1 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:

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

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 $\sqrt{N_Q}$ geteilt. Dieser Term wird von 1 abgezogen. Eine User*in für $\tilde{q}_P = \tilde{q}_U$ einen Post maximal gut.

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

In Abbildung ?? werden anhand von jeweils zwei Beispieldaten, $P1, P2$, und User*innen, $U1, U2$, 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 $U1$.

4 User*innenmeinung

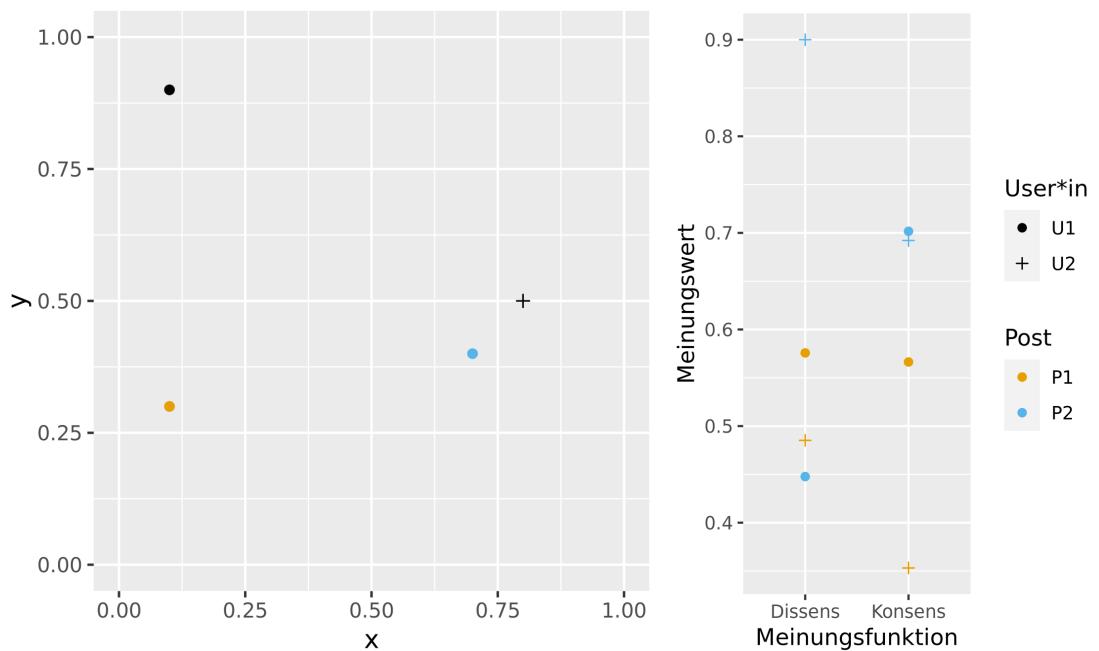


Figure 4.1: Vergleich der beiden Meinungsfunktionen Konsens und Dissens

5 Bewertungsmetriken

Mit Bewertungsmetriken sollen Posts anhand ihrer Relevanz sortiert werden. Ein Post P erhält durch eine Bewertungsmetrik den Score $s_P = B(P)$ zugewiesen. Die Posts werden anhand des zugewiesenen Scores sortiert und auf der Plattform zur Verfügung gestellt.

Die wahre Relevanz R_P steht in der Berechnung der Bewertungsmetrik nicht zur Verfügung, daher müssen die "sichtbaren" Parameter von P verwendet werden um die Relevanz anzunähern.

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

Eine ideale Bewertungsmetrik ist fair, 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 $w_{P_i} > 1$, er wurde von mindestens einer*r User*in wahrgenommen

5.1 Bewertungstransformation

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

Für den zweidimensionalen Fall mit Up- und Downvotes (u_P, d_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:

$$v_{diff}(b_P) = u_P - d_P \quad (5.1)$$

Die Differenz wird im Reddit Hot Ranking verwendet.

5.1.2 Anteil

Berechnet wird der Anteil der Upvotes zur Gesamtzahl der Votes:

$$v_{anteil}(b_P) = \frac{u_P}{u_P + d_P} \quad (5.2)$$

5.1.3 Wilson Score

Wie in [Online Paper über Wilson] 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 $n_P = u_P + d_P$, dem Punktschätzer $\hat{p}_P = u_P / n_P$ und dem Quantil c_α der Normalverteilung zum Irrtumsniveau α ergibt sich in Formel 5.3 der Wilson Score:

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

5.2 Bewertungsmetriken

Bewertungsmetriken berechnen den Score $S_{P,t}$ für einen Post unter Betrachtung der Postparameter eines Postes P zum Modellschritt t .

5.2.1 Hacker News Ranking

Hacker News verwendet eine Metrik, in der die Upvotes u_P eines Postes ins Verhältnis zum Alter a_P in Stunden zum Zeitpunkt t setzt. Somit besitzt der Bewertungsraum die Dimension 1. Das Alter wird mit der Gravitätikonstanste $G = 1.8$ potenziert. Die Bewertungstransformation lautet $v(b_P) = u_P - 1$ so ergibt sich die Hacker News Bewertungsmetrik in Formel 5.4:

$$S_{P,t} = \frac{u_{P,t} - 1}{(a_{P,t} + 2)^G} \quad (5.4)$$

5.3 Verallgemeinertes Hacker News Ranking

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

$$S_{P,t} = \frac{v(b_{P,t})}{(a_{P,t} + 2)^G} \quad (5.5)$$

5.4 Reddit Hot Ranking

Im Reddit Hot Ranking fließt die evaluierte Voteanzahl d_P eines Postes logarithmisch, und der Zeitpunkt der Veröffentlichung r_P einfach ein.

Um den Veröffentlichungszeitpunkt e_P zu messen, wird die Differenz in Sekunden von diesem zum Zeitpunkt E am 8.12.2005 um 07:46:43 in Formel 5.6 berechnet.

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

Die Default Bewertungstransformation des Reddit Hot Ranking ist die Differenz aus Up- und Downvotes:

$$v(b_P) = v_{diff}(b_P) \quad (5.7)$$

Der Parameter z wird in 5.8 auf das Maximum des Betrages von $v(b_{P,t})$ aus Formel 5.7 und 1 gesetzt.

$$z_{P,t} = \begin{cases} |v(b_{P,t})| & \text{falls } |v(b_{P,t})| \geq 1 \\ 1 & \text{sonst} \end{cases} \quad (5.8)$$

Es ergibt sich die Bewertungsmetrik des Reddit Hot Rankings in Formel 5.9:

$$S_{P,t} = sign(v(b_{P,t})) * log_{10}(z_{P,t}) + \frac{e_P}{4500} \quad (5.9)$$

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 $v(b_P)$, 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 w mit ein. Daraus ergibt sich die Bewertungsmetrik in 5.10:

$$S_{P,t} = \frac{\frac{b_{P,t}}{w_{P,t}+1}}{(a_{P,t} + 2)^G} \quad (5.10)$$

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:

$$S_{p,t} = \frac{v(b_{P,t}) - S_{p,t-1}}{(a_{p,t} + 2)^G} \quad (5.11)$$

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 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:

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

5.7.2 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 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 [Biega2018405](#) dient zur Berechnung der Güte von Bewertungsmetriken. Der DCG in Formel 6.1 bestraft Bewertungsmetriken, welche Posts mit hoher Qualität auf hintere Rankingpositionen einordnet. Die Qualität eines Posts $Q_{P,i}$ fließt durch den Logarithmus der Rankingposition des Posts ins Verhältnis gesetzt in den DCG ein.

$$DCG(B) = \sum_{i=1}^N \frac{2^{Q_{P,i}} - 1}{\log_2(i + 1)} \quad (6.1)$$

Normalized Discounted Cumulative Gain Der nDCG normalisiert den DCG einer Bewertungsmetrik mit einer idealen Bewertungsmetrik B_I , welche die Posts absteigend nach Qualität 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 Lerman2014 und Salganik2006854 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. 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 $G = 1$ maximal. Der Gini-Koeffizient wird beschrieben durch:

$$G = \frac{1}{2S \sum_{i=1}^N v_i} \sum_{i,j} |v_i - v_j| \quad (6.3)$$

Top-k Gini-Koeffizient Der Top-k Gini-Koeffizient berechnet den Gini-Koeffizient für die k qualitativ besten Posts.

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 N_i 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:

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

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:

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

6 Evaluationsmethode

wobei $1_{v_{P_i}} = 1$, 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.jl¹ für Julia². Die verwendete Version von Agents.jl ist ein weitereintwickelter Fork der Verion 3.0.0.

Sämtliche Funktionalität ist in dem Paket VotingProtocols³ zusammengefasst.

7.1 agentenbasierte 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 Algorithmus ?? beschrieben. 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. In dieser Funktion wird die Bewertungsmetrik angewendet und die Posts entsprechend angeordnet.

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
end for
```

7.1.1 Agent*innenschrittfunktion

Die Agent*innenschrittfunktion wird in Algorithmus ?? beschrieben. Für ein*e User*in U wird zuerst berechnet ob sie in der aktuellen Iteration aktiv ist. Dies wird anhand der Aktivitätswahrscheinlichkeit von a_U berechnet. Ist U aktiv werden so viele Posts von U

¹agent.jl

²julialang.org

³github adresse

auf der Platform betrachtet wie durch die Konzentration c_U vorgegeben sind. Für jeden Post P bildet sich U mit der User*innenratingfunktion eine Meinung $r_{U,P} = R(P, U)$ und bewertet ihn möglicherweise.

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 Algorithmus ?? dargestellt. Für jeden Post P wird mit der Bewertungsmetrik $S_{P,t} = S_t(P)$ berechnet. Anschließend werden neue Posts dem Modell hinzugefügt. Die Anzahl der neuen Posts ist durch den Modellparameter `new_posts_per_step` 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 eigen Modellkonfiguration erstellt und simuliert. Die Modellkonfigurationen werden durch eine Liste aus Tupeln definiert.

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

Figure 7.1: Beispiel Modellkonfigurationen

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

Der erste Eintrag der Tupels definiert die Grundkonfiguration. 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 der `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 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. Diese 18 Modelle können nun berechnet werden.

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 die berechnete DataFrame der Iterationsevaluationsmethoden.

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

Die Iterationsevaluationsfunktionen `ndcg` und `gini` berechnen für jede Modelliteration den *nDCG* bzw. Gini-Koeffizienten *G* der Bewertungsmetrik. Durch das Macro `@gini_top_k(100)` wird eine Funktion `gini_top_100` erzeugt, welche den Gini-Koeffizienten für die 100 besten Posts berechnet.

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` kön-

nen 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.

```

1 \label{lis:eval}
2 iter_eval_functions =
3 [
4   ndcg,
5   gini,
6   @gini_top_k(100)
7 ]
8
9 model_eval_functions =
10 [
11   posts_with_no_views,
12   @area_under(:ndcg),
13   @area_under(:gini),
14   @area_under(:gini_top_100),
15   @model_parameter(:init_score),
16   @model_parameter(:scoring_function),
17   @model_parameter(:relevance_gravity),
18   @model_parameter(:user_rating_function),
19 ]

```

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.

⁴RFOTTNOT

8 Ergebnisse

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

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

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

Vorst werden Modellkonfigurationen unter den vier Fällen mit jeweils der Relevanzgravität $G_R = 0,2$ und den User*innenmeinungsfunktionen Konsens R_K und Dissens R_D ausgewertet. In Abbildung ?? links sind $T(G)$, $T(nDCG)$, und $P_{v=0}$ von Modellen nach der Konfiguration 1 in die vier Fälle eingeteilt. Grau hinterlegt sind sämtliche Modelle, farblich hervorgehoben die Mittelwerte der unterschiedlichen Modellkonfigurationen. Farblich unterschieden wird zwischen den Bewertungsmetriken. Aus der Korrelationsmatrix rechts in der Abbildung geht hervor, dass ρ der Modelle für die Fälle mit $G_R = 2$ und der Fall $R = R_K$ und $G_R = 2$ stark durch den Spearman-Koeffizienten korreliert sind.

Im Folgenden werden nur noch die Fälle $R = R_K$ und $G_R = 0,2$ betrachtet, die drei stark korrelierten Fällen müssen nicht einzeln untersucht werden.

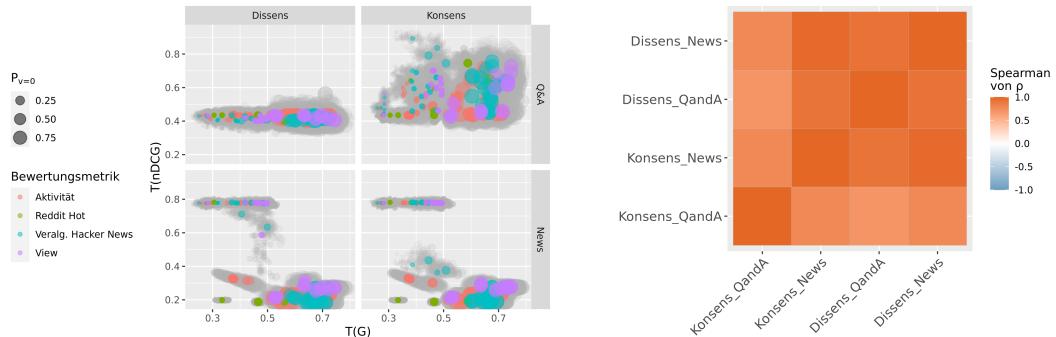


Figure 8.1: Fälle von Plattformen

8.1 Größe des Bewertungsvektor

In Abbildung ?? ist die Modellsimulation mit Konfiguration 1 der Fälle $R = R_K$ und $G_R = 0,2$ zu sehen . Farblich markiert ist die Größe des Bewertungsvektors. In 64.4% der Konfigurationen wird mit $B_N = 2$ ein besseres ρ erzielt.

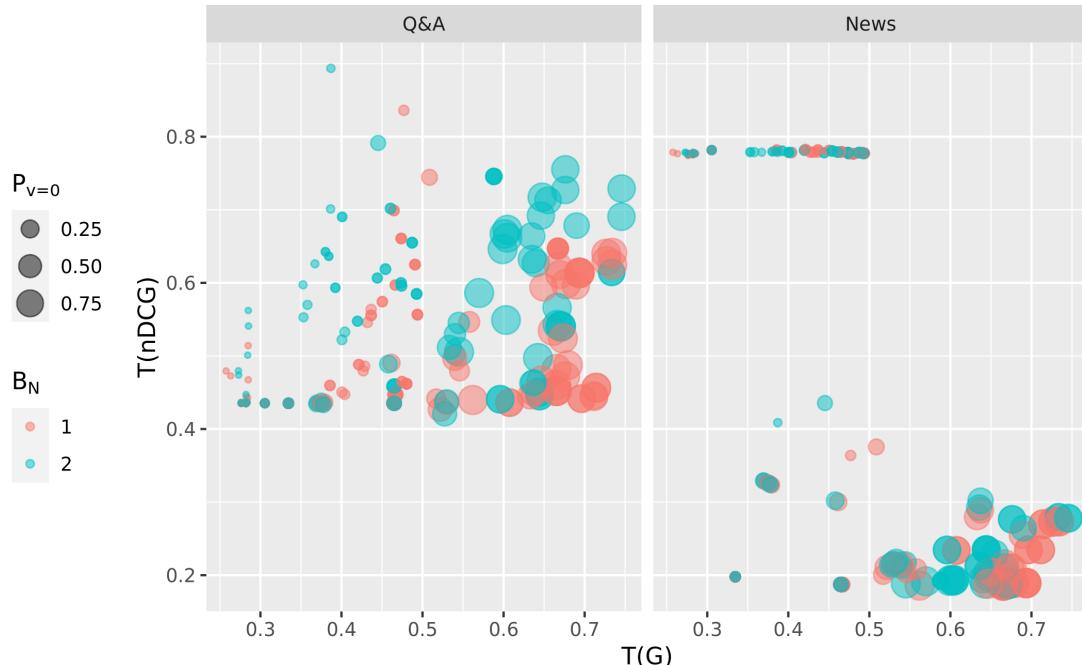


Figure 8.2: Bewertungsvektor

8.2 Initialscore

Abbildung ?? zeigt ρ der Konfiguration 1 der beiden beiden Fälle. Auf der x-Achse sind die Bewertungsmetriken aufgetragen, farblich markiert ist der Initialwert. $s_0 = 0$ ist für alle Modellkonfigurationen die schlechteste Wahl.

Nach Konfiguration 3 ist der Einfluss auf die Bewertungsmetriken des Initialscores dargestellt. In der Konfiguration des verallgemeinerten Hacker News Metrik wird durch die Erhöhung von s_0 $T(G)$ verringert und $T(nDCG)$ erhöht. Für $s_0 > 70$ verringert sich $T(nDCG)$ signifikant. Bei der Aktivitätsmetrik führt eine Erhöhung von s_0 zum Abnahme von $P_{v=0}$, $T(nDCG)$ und $T(G)$. In der Viewmetrik wird zwischen $s_0 \in [10, 30]$ $T(G)$ und $P_{v=0}$ reduziert, für $s_0 > 30$ wird hauptsächlich $T(nDCG)$ reduziert. Für die Reddit Hot Metrik ist $s_0 = \{0, 30000\}$. Für s_0 erhalten neue Posts einen höheren Initialscore, als jemals von der Metrik zugewiesen wird. Der hohe Initialwert liefert bessere kleinere $T(G)$, $T(nDCG)$ und $P_{v=0}$.

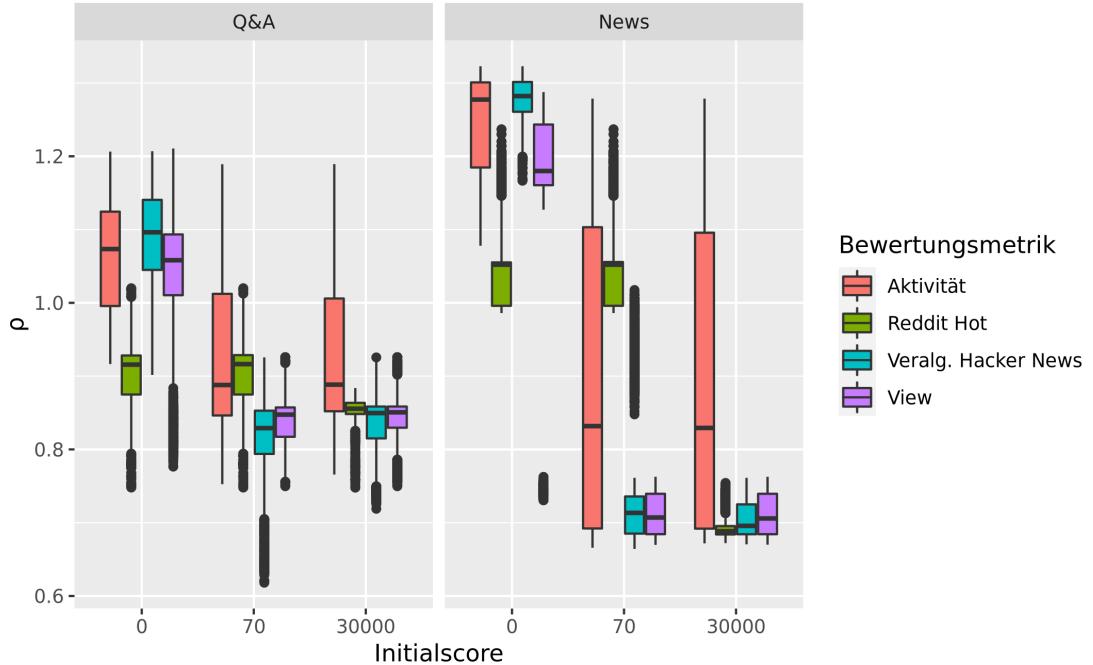


Figure 8.3: Initialscore

Die Variierung des Initialscores wirkt sich auf unterschiedliche Art auf die unterschiedlichen Bewertungsmetriken aus.

8.3 Gravität

Der Einfluss der Gravität der drei Bewertungsmetriken mit Gravität wird nach Konfiguration 4 in Abbildung ?? gezeigt. Die Aktivitätsmetrik besitzt bei $G_R = G = 0$ den Wert $P_{v=0} = 0.84$. So wird bei der Aktivitätsmetrik das geringste ρ mit $G = 0$ erzielt. Für die View- und Hacker News Metrik steigt ρ mit der Erhöhung von G im Fall $G_R = 0$. Für den Fall $G_R = 2$ wird das kleinste ρ mit $G = 0$ für die Bewertungsmetriken Aktivität und Verallgemeinertes Hacker News erzielt. Auf die Viewmetrik hat in diesem Fall die Variierung von G keinen signifikanten Einfluss.

8.4 Bewertungstransformation

In Abbildung ?? ist farblich die verwendete Bewertungstransformation gekennzeichnet. Ein Datenpunkt beschreibt den Mittelwert einer Modellkonfiguration. Es zeigt sich, dass v_{diff} in den meisten Bewertungsmetrikfällen eine größere Varianz bezüglich ρ als v_{anteil} und v_{wilson} im Fall $R = R_K$ und $G_R = 0$ besitzt v_{diff} jedoch einen geringeren Mittelwert.

8 Ergebnisse

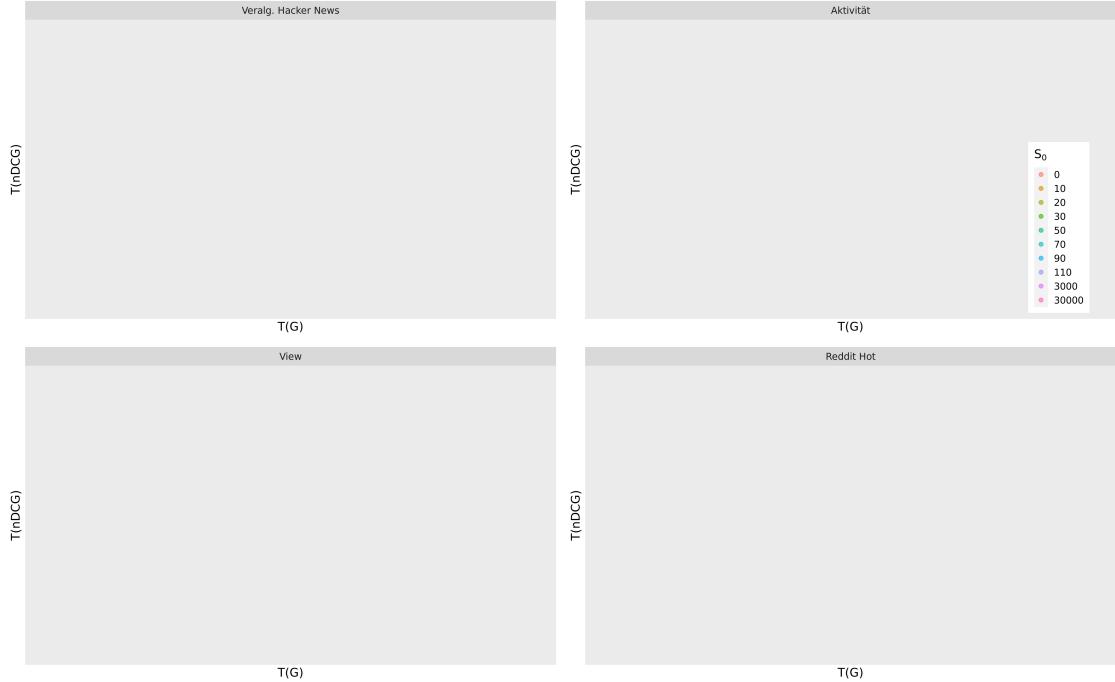


Figure 8.4: Initialscore in Bewertungsmetriken

Im weiteren betrachteten Fall $R = R_K$ und $G_R = 0$ ist die Performance von v_{anteil} und v_{wilson} sehr ähnlich und meist besser als v_{diff} .

8.5 Zufallsbewertung

Die Boxplots in Abbildung ?? zeigen, dass eine zufällige Abweichung bei allen Metriken außer der verallgemeinerten Hacker News im Fall $G_R = 0$, zu einer signifikanten Verkleinerung von $P_{v=0}$ und $T(G)$ führt. Dabei führt die σ -Abweichung zu einer stärkeren Reduktion. Auch $T(nDCG)$ wird bei $G_R = 0$ verringert, wieder stärker durch die σ -Abweichung. Die Hacker News Metrik reagiert in diesem Fall gering auf die Zufallsabweichung. Im Fall von $G_R = 2$ wirkt sich die Zufallsabweichung nicht signifikant aus.

8.6 Modellparameter

8.6.1 Iterationslänge

Wird die Iterationslänge M_N variiert hat dies Einfluss auf das Ergebnis. In Abbildung 8.8 links oben sind die vier Bewertungsmetriken nach Konfiguration 4 nach $G_R = 1, 2$ dargestellt. Im Fall $G_R = 2$ zeigt sich, dass durch Erhöhung der Iterationsanzahl

8 Ergebnisse

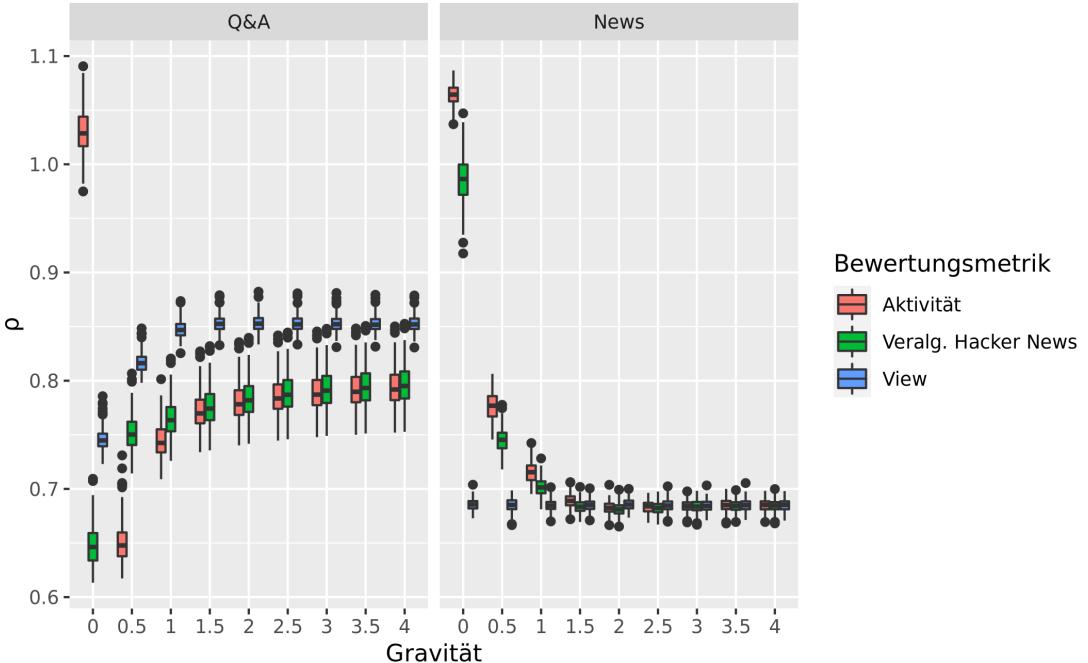


Figure 8.5: Gravität

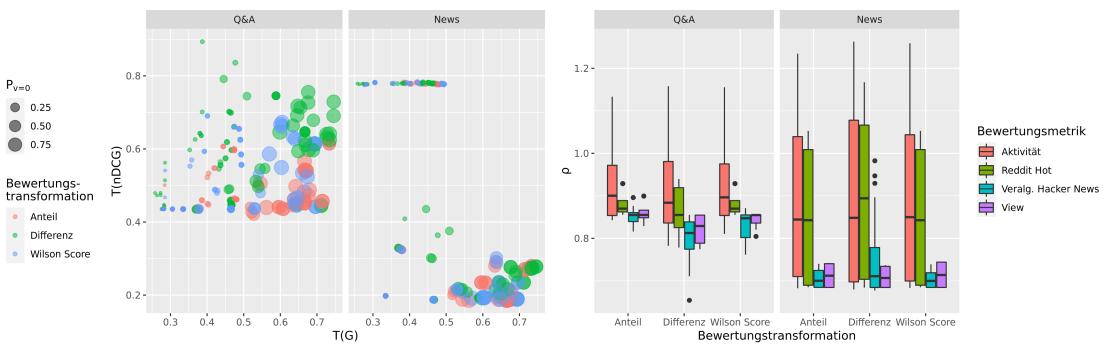


Figure 8.6: Bewertungstransformation

$T(nDCG)$ und $P_{v=0}$ verkleinert wird. Für den Fall $G_R = 0$ zeigen die drei weiteren Plots die Entwicklung der drei Evaluationsparameter mit steigender Iterationslänge. $T(nDCG)$ wird für die View- und Aktivitätsmetrik größer. Die Varianz von $T(nDCG)$ verringert sich für alle Metriken außer der Hacker News Metrik. $P_{v=0}$ 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.

8 Ergebnisse

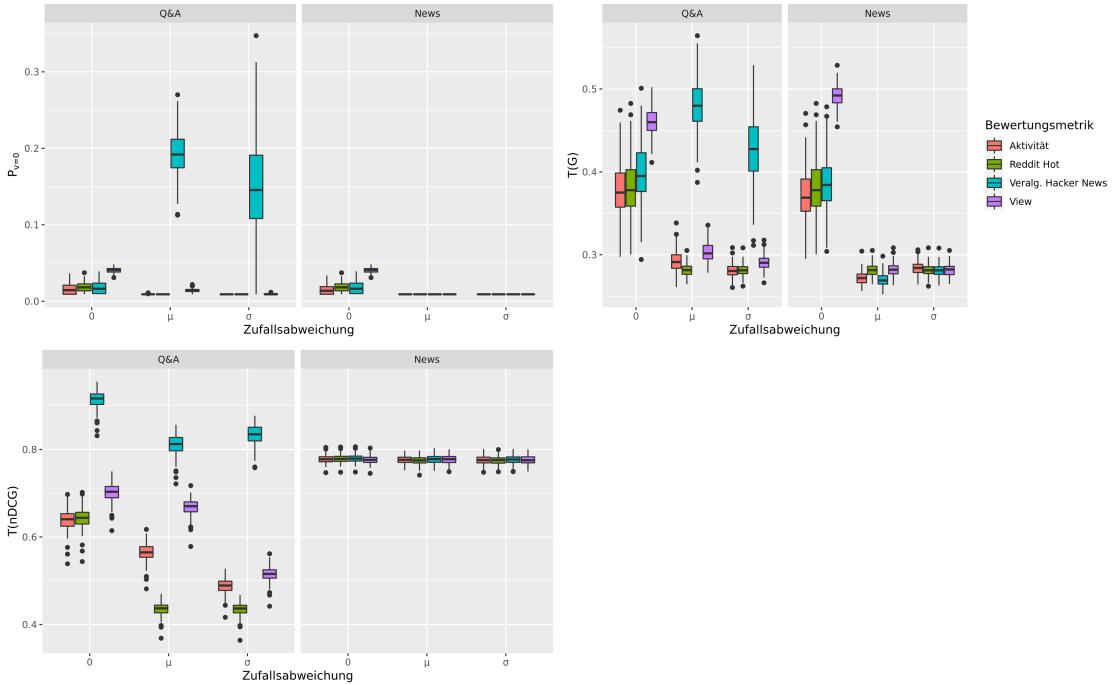


Figure 8.7: Zufallsbewertung mit Konfiguration 4

8.6.2 Qualitätsraum

Der Einfluss der Größe des Qualitätsraum Q_N wird in Abbildung ?? untersucht. Auf den Fall $G_R = 0$ hat die Größe nur geringfügig Einfluss, wie auch der Boxplot rechts bestätigt. Im Fall $G_R = 2$ erhöht sich mit Q_N $T(nDCG)$. Auch die Varianz von $T(nDCG)$ verringert sich.a

8.6.3 User*innen- und Postanzahl

In Abbildung ?? ist der Einfluss der Anzahl der Startposts nach der *speziellen Modellkonfiguration* dargestellt. 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 schon zu Beginn erfassen können. Die Folge ist die markante Reduzierung von $T(G)$ und $T(nDCG)$. 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. Trotzdem ist davon auszugehen, dass mit einer hohen Wahl der Parameter die Realität besser modelliert wird.

8 Ergebnisse

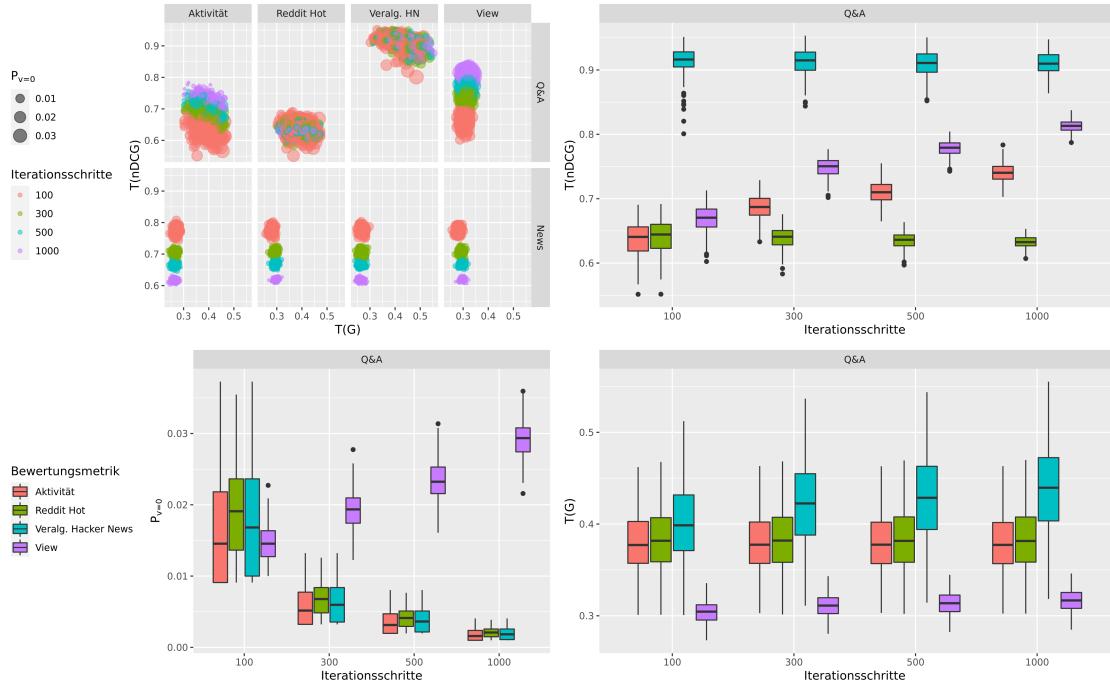


Figure 8.8: Unterschiedliche Iterationslängen

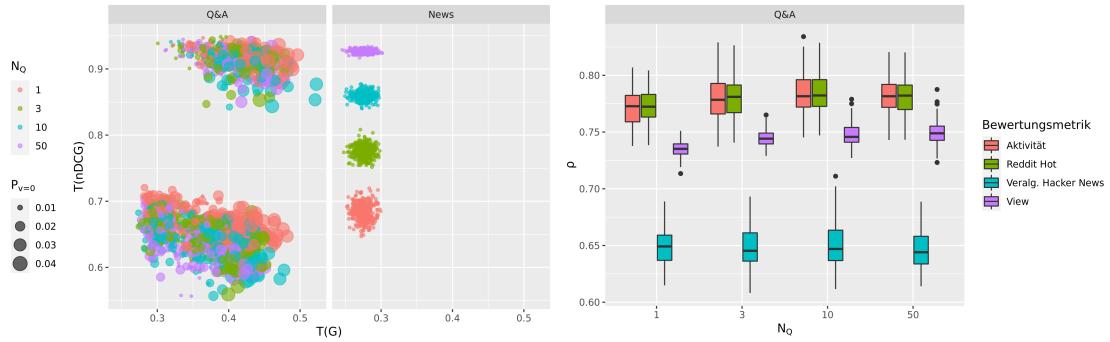


Figure 8.9: Qualitätsdimensionen

8.6.4 Vorsortierung der Posts

8.7 User*innenparameter

Nach der *speziellen Modellkonfiguration* sind Simulationsergebnisse in Abbildung 8.11a dargestellt. Die Verteilungen der User*innenaktivität und der Bewertungszufriedenheit werden variiert. In Abbildung 8.11b sind die Dichteverteilungen der vier verwendeten Verteilungen dargestellt. Wie an der Viewmetrik im Fall $G_R = 0$ zu sehen ist führt die Variierung der Verteilung der Bewertungsverteilung zur Verschiebung des Mittelwerts

8 Ergebnisse

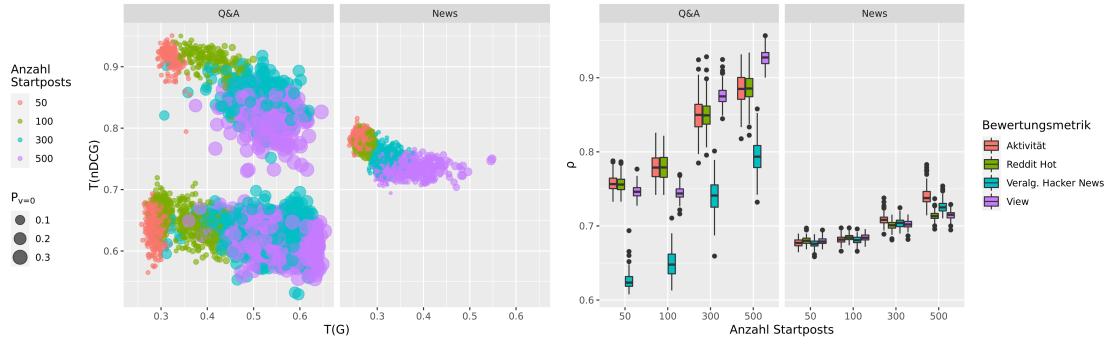
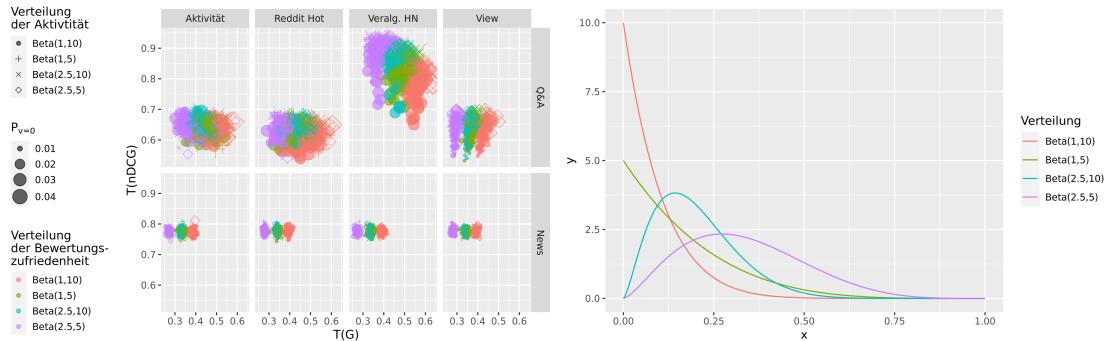


Figure 8.10: Startposts

von $T(G)$, während die Variierung der Aktivitätsverteilung zur Verschiebung des Mittelwerts von $P_{v=0}$ und $T(nDCG)$ führt. Für den Fall $G_R = 2$ hat die Variierung der Aktivitätsverteilung keinen signifikanten Einfluss. Die Simulationsergebnisse, welche durch die Verwendung unterschiedlicher Verteilungen entstanden sind stehen in starker Korrelation. Die Ergebnisse einzelner Modellkonfigurationen ist weitestgehend unabhängig von der Wahl der Verteilungen der Bewertungszufriedenheit und der Aktivität.



(a) Variierung der Verteilung von Aktivität und (b) Dichteverteilungen der verwendeten β -Verteilungen

Figure 8.11: User*innenparameter

Für die Konzentrationsverteilung ergibt sich das gleiche Bild. Wird der Erwartungswert der Konzentrationsverteilung erhöht, so verringert sich $T(G)$.

Wird der Erwartungswert der poisson-verteilten Konzentration erhöht, so verringert 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. Durch die Wahl der Konzentrationsverteilung wird ebenso wenig das Ergebnis einzelner Modellkonfigurationen beeinflusst.

9 Diskussion der Ergebnisse

Alle getesteten Bewertungsmetriken erzeugen in der gleichen Umwelt unterschiedliche Ergebnisse. Die Wahl der Bewertungsmetrik hat Einfluss auf die Fairness der Sortierung der Posts.

Die in [Luu] vorgeschlagene zufällige Abweichung führt in dieser Simulation zur Verbesserung von Metriken auf News- und Diskussionsplattform. Der aggregierte Gini-Koeffizient $T(G)$ kann signifikant reduziert werden, während der aggregierte nDCG $T(nDCG)$ stabil bleibt.

In [Wilson Score Paper] wird der Wilson-Score vorgeschlagen um Up- und Down-votes 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 Plattform auf der Konsens herrscht und die Posts nicht an Relevanz verlieren, wie in einer technischen Q&A Plattform ist die Differenz-Bewertungstransformation überlegen.

Bewertungsmetriken liefern bessere Ergebnisse, wenn Posts mit einem Initialscore $s_0 > 0$ ausgestattet werden. Die Wahl des besten Initialscores ist von der Bewertungsmetrik abhängig. Für die Reddit Hot Metrik konnte der optimale Initialscore nicht 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.

In dieser Arbeit wurde der sehr große Featureraum nicht vollständig exploriert. Einzelne Metrikparameter wie die Gravität 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.

Einfluss der Modell- und User*innenparameter Die Vergleichbarkeit der Ergebnisse ist nicht unmittelbar abhängig von der Wahl der artifiziell erzeugten 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 der betrachteten Ergebnisparameter. Im Fall von $G_R = 0$ reagieren nicht alle Bewertungsmetriken gleich auf die Veränderung der Iterationsanzahl. Dies führt zu einer Verzerrung der Simulationsergebnisse nach Iterationslänge.

9 Diskussion der Ergebnisse

Wird die Anzahl der Posts und User*innen variiert, hat dies 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 sind die User*innen Postzahlen um ein Vielfaches größer als die in der Simulation verwendeten. Es ist davon auszugehen, dass sich die beschriebenen Effekte mit weiter ansteigender User*innen und Postanzahl verstärkt werden.

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.

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 $B_N = 1,2$, Modelle welche User*innen Posts nur positiv oder positiv und 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.

Die Framework bietet die einfache Definition von neuen Bewertungsmetriken und deren Parameter. In [Masterarbeit Felix] 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.

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.

Die verwendeten User*innenmeinungsfunktionen sind sehr einfach und können nur sehr eingeschränkt die Realität modellieren. Das Alter des Posts fließt ebenfalls nicht in die Meinungsfunktion ein. In zukünftigen Arbeiten können die Meinungsfunktionen weiter ausgefeilt werden.

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.

– Moderierte Systeme

Formelzeichen

M	Modell
P	Post
U	User*in

Abkürzungsverzeichnis

M Model