1 Einleitung

1.1 Bewertungsdynamiken

Durch immer weiter fortschreitende Ausbreitung des Internets, wächst die Anzahl der User*innen von sozialen Medien und anderen Onlineplattformen, damit einhergehend die Anzahl der veröffentlichten Beiträge. Es werden so viele Beiträge veröffentlicht, dass es nicht möglich ist allen User*innen alle Beiträge anzuzeigen. Es werden bewertungsmetrische Algorithmen eingeführt, welche Beiträge bewerten und anschließend filtern und vorsortieren. Im Idealfall werden durch die angewendeten Algorithmen schlechte Beiträge, weil sie z.B. diskriminierend, langweilig oder irrelevant sind, aussortiert und nur wenigen User*innen angezeigt, jedoch gut und relevante Beiträge vielen User*innen sichtbar gemacht werden.

Onlineplattformen verwenden unterschiedliche Votingsysteme mit welchen User*innen Beiträge bewerten können und anhand derer Bewertungsmetriken Beiträge sortieren. Hacker News[1] erlaubt User*innen Posts nur positiv, mit Upvotes, zu bewerten. Auf Reddit[2] ist es ebenfalls möglich Posts mit Downvotes runterzuwerten. Onlineshops wie TheCubicle[3] verwenden ein 1 bis 5 Sterne System um Artikel zu bewerten.

Wie in [Page quality: In search of an unbiased web ranking] für Websuchalgorithmen beschrieben unterliegen viele Bewertungsmetriken dem *rich-get-richer*-Effekt, welcher dem Idealfall entgegen wirkt. Der *rich-get-richer*-Effekt führt dazu, dass Beiträge welche viel Aufmerksamkeit und dadurch viel Aktivität in Form von Kommentaren oder Bewertungen erhalten werden weiteren User*innen angezeigt. Beiträge hingegen mit wenig Aufmerksamkeit erhalten auch zukünftig wenig Aktitvität, weil sie von wenigen User*innen wahrgenommen werden. Wie in [Description and Prediction of Slashdot Activity] gezeigt wird ist es entscheidend wie viel Aktivität ein Beitrag in einer kurzen Zeitspanne erzeugen kann. Ist es viel Aktivität wird der Beitrag insgesamt viel Aufmerksamkeit erhalten, ist es jedoch wenig Aktivität wird der Beitrag auch insgesamt wenig Aufmerksamkeit erfahren.

Es ist wünschenswert eine Bewertungsmetrik zu finden, welche *rich-get-richer*-Effekte minimiert und qualitativ hochwertige Beiträge sichtbarer macht als qualitativ minderwertige Beiträge.

[1] `news.ycombinator.com`
[2] `reddit.com`
[3] `thecubicle.com`

Es ist nicht ausgeschlossen, dass die Qualität einer Bewertungsmetrik vom gewählten Votingsystem abhängig ist, daher soll bei der Suche nach einer optimalen Bewertungsmetrik das Votingsystem einbezogen werden.

In dieser Arbeit wird ein agentenbasiertes Modell entwickelt um unterschiedliche Bewertungsmetriken auf der Grundlage unterschiedlicher Votingsysteme zu vergleichen.

1.2 agentenbasierte Modellierung

In der agentenbasierten Modellierung werden die Aktionen und Interaktionen vieler Entitäten, den Agenten, simuliert. Die Agenten erhalten Verhaltensregeln, durch ihre Aktivtät kann ein komplexes System beschrieben werden. Die Auswirkungen auf das System können betrachtet werden.

Wie [Agents.jl] beschreibt können viele komplexe Systeme nicht vollständig durch herkömmliche mathematische Methoden beschrieben werden. Komplexe Systeme hängen von dem Verhalten der Elemente, der Agenten, des Systems ab. Kleine Änderungen am Verhalten der Agenten können große Veränderungen des Gesamtsystems hervorrufen. Durch agentenbasierte Modellierung können nicht lineare Modelle beschrieben.

Agenten können in einem Raum angeordnet werden, sodass sie mit ihren Nachbarn interagieren, wie im Modell von [Shelling], oder als Netzwerk indem jeder Agent mit jedem Agenten interagieren kann. Möglich ist auch, dass die Agenten nicht untereinander, nur mit dem System, interagieren.

Die Popularität steigt stetig, agentenbasierte Modelle sind beliebt um unter anderem biologische, ökonomische und soziale Systeme zu modellieren.

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 neue 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 beinflussen 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*innnen 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 Bewertungsmetrik nicht optimieren lässt. Sie stellen ein Optimierungsproblem auf, welche durch die Anordnung von unterschiedlichen Bewertungsmetriken 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 Bewertungsmetriken gerankt werden:

1. Zufall: in zufälliger Reihenfolge
2. Popularität: nach der Anzahl der Empfehlungen sortiert
3. Aktitvität: nach der Zeit der letzten Empfehlung sortiert
4. Fixierung: stets in gleicher Reihenfolge
5. Fixierung invertiert: in umgekehrter fixierter Reihenfolge

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 Platformen abzuschätzen.

In [Description and Prediction of Slashdot Actitvity] wird versucht die Aktitvität (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 unteruscht 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 auszuglätten.

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 ein 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

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 Qualität

User*innen von sozialen Medien können Posts nach unterschiedlichen Kriterien und Qualitätsmerkmalen betrachten. Auf einer News Plattform achten User*innen auf die Informativität und Aktualität. Auf einer Fotoseite werden Beiträge vielleicht eher nach Ästhetik und Originalität bewertet. Auf welche und wie viele Qualitätsmerkmale die User*innen letztlich achten ist nicht bekannt.

Qualität wird über 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 Objekt 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 Qualiätsmerkmale können so einfach in Korrelation gesetzt werden.

3.2 Bewertungraum

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

Im folgenden sind die Parameter der im Modell verwendeten Posts beschrieben.

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 (Abschnit 5 zugeordnet wurde.

Der Initialscore s_0 aller Posts kann varriert werden.

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

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.

Es wird keine Unterscheidung zwischen Darstellungsqualität wie in [Felix masterarbeit] gemacht. Diese ist hoch, wenn auf den ersten Blick deutlich wird um was sich der Post handelt. Alle User*innen erfassen auf den ersten Blick den gesamten Qualitätsumfang der Posts.

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 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:

3.1:

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

3.4 Bewertungsmetriken

Bewertungsmetriken sollen Posts anhand ihrer Relevanz sortieren. Da die wahre Relevanz von Posts bei der Verwendung von Bewertungsmetriken nicht bekannt ist, muss diese durch die weiteren Postparameter approximiert werden.

Das Kapitel 5 beschäftigt sich ausführlich mit Bewertungsmetriken.

3.4.1 Vorsortierung der Posts

Bei der Simulation einer Kommunikationsplatform besteht die Problematik des *Cold Starts*. Diese besagt, dass es in einer realen Umgebung den Zeitpunkt 0 der Simulation nicht gibt. Zum Zeitpunkt 0 der Simulation sind die Posts zufällig angeordnet. Kein Posts wurde betrachtet oder bewertet.

Um dem *Cold Start* entgegen zu wirken können die Posts nach Qualität vor der Simulation sortiert werden. Der Sortierungskoeffizient ω befindet sich im Intervall $[-1, 1]$. Bei $\omega = 0$ findet keine Vorsortierung statt. Bei $\omega = 1$ sind die Posts vollständig nach Qualität sortiert, bei $\omega = -1$ sind die Posts vollständig umgekehrt nach Qualität sortiert.

3.5 User*innen

Um ein agent*innenbasiertes Modell einer Onlinekommunikationsplattform zu entwickeln ist es elementar das User*innenverhalten zu beschreiben. Leider stehen im Rahmen dieser Arbeit keine Daten zur Verfügung, an denen es möglich ist User*innnenparameter zu fitten.

Die Parameter werden stattdessen durch plausible Verteilungen approximiert.

3.5.1 Aktivtität

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

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.

[Bild mit unterschiedlichen β-Verteilungen]

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 und wird über eine Poisson-Verteilung modelliert.

[Bild mit unterschiedlichen Poissonverteilungen??]

3.5.3 Qualitätsperzeption

Durch die Qualitätsperzeption wird festgelegt wie sehr 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 User*innenmeinungsfunktion

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 $R(P,U) : \mathbb{R}^{n_Q} \to \mathbb{R}$ 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.

3.6 User*innenmeinungsverteilung

Werden die Meinungswerte vieler User*innen bezüglich vieler Posts berechnet, dann spannen diese eine Wahrscheinlichkeitsverteilung W auf. Diese wird im Modell dazu verwendet zu entscheiden, ob User*innen Posts bewerten. (Siehe dazu Abschnitt 3.6.1).

Die User*innenmeinungsverteilungfunktion in 3.2 wird empririsch berechnet. Dazu werden aus der Qualitätsverteilung jeweils n Pseudoposts Ps und Pseudouser*innen Us erzeugt. Diese werden über die Meinungsfunktion R kombiniert. $1_{\{x \leq R(Ps_i, Us_j)\}}$ ist 1 für $x \leq R(Ps_i, Us_j)$ und sonst 0.

$$F_W(x) = \frac{1}{n^2} \sum_{i,j=1}^{n} 1_{\{x \leq R(Ps_i, Us_j)\}} \tag{3.2}$$

3.6.1 Bewertungszufriedenheit

Im Modell wird angenommen, dass User*innen Posts nur bewerten, wenn der berechnete Meinungswert über bzw. unter einen bestimmten Grenzwert fällt. Dieser Grenzwert wird mithilfe der Bewertungszufriedenheit β_U definiert.

Der Grenzwert wird mit der Umkehrfunktion der User*innenmeinungsverteilungsfunktion $g = F_W^{-1}(\beta_U)$ berechnet. g ist somit das β_U-Quantil der empirischen Dichteverteilung von M

Sobald $m_{P,U} < g$ bzw. $m_{P,U} > g$ wird der Post P von der User*in U bewertet.

Die Bewertungszufriedenheit ist kontinuierlich auf dem Intervall $[0, 1]$ verteilt und wird mit einer β-Verteilung modelliert.

3.6.2 Extreme User*innen

Extreme User*innen besitzen eine sehr starke Meinung. Die Qualiätsperzeption ist $q_{U_E} = q_P + E$. Wobei E angibt, wie extrem die Meinung ist. Extreme User*innen sind in jedem Modellschritt aktiv und bewerten jeden Post: $a_{U_E} = \beta_{U_E} = 1$. Außerdem besitzen extreme User*innen eine sehr hohe Konzentration: $k_{U_E} = 500$.

E_M ist eine Matrix, Anzahl und Art der extremen User*innen definiert. In Formel 3.3 ist eine Beispielmatrix gezeigt.

$$E_{MBsp} = \begin{pmatrix} 0.02 & 20 \\ 0.05 & -10 \end{pmatrix} \tag{3.3}$$

In der ersten Spalte befindet sich der Anteil an der gesamten User*innenzahl mit dem E-Wert in der zweiten Spalte. In diesem Beispiel sind 2% extreme User*innen mit $E = 20$ und 5% extreme User*innen mit $E = -10$. Die restlichen 93% User*innen sind "normal".

3.7 User*innen und Postanzahl

In der folgenden Tabelle sind die Modellparameter der User*innen- und Postanzahl angegeben:

N_U	Anzahl der User*innen
N_{Pstart}	Anzahl der Posts zu Beginn der Simulation
N_{Piter}	Anzahl der neuen Posts pro Iteration
N_P	Gesamtanzahl der Posts, $N_P = N_{Pstart} + i_M * N_{Piter}$

4 User*innenmeinung

Für eine User*in U die einen Post P betrachtet ist die Meinungsfunktion $R(P, U) : \mathbb{R}^{N_Q} \to \mathbb{R}$, 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 beachten wird eine Transormation der Qualitätsparameter vollzogen, welche diese auf das Intervall $[0, 1]$ begrenzen. Die Transformation wird durch die logistische Funktion l in Formel 4.1 vollzogen:

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

Somit ergeben sich die transformierten Qualitäts- bwz Qualitätsperzeptionsvektoren der Post und User*innen durch die komponentenweise Anwendung von s auf q_P und q_U

$$\tilde{q}_P = l(q_P) \tag{4.2}$$
$$\tilde{q}_U = l(q_U) \tag{4.3}$$

4.2 Approxamtion des User*innenratings

Im folgenden werden zwei Ansätze eingeführt um die User*innenmeinungsfunktion $R(P, U)$ zu approximieren.

4.2.1 Konsensrating

Im Konsensrating wird davon ausgegangen, dass User*innen "gute" Posts eher als diese erkennen. Alle User*innen sind sich über die besonders guten und schlechten Posts einig

und erkennen diese richtig. Bei durchschnittlichen Posts kann es auch zu Meinungsverschiedenheiten kommen, je nach der Qualitätsperzeption der User*innen.

Das Konsensrating ist eher auf einen Q&A Platform oder einem Techforum wie Hacker News denkbar. Konstruktive Beiträge sind eher von destruktiven zu unterscheiden. User*innen die besonders gut in einem Thema sind haben an Beiträge zu diesem Thema möglicherweise höhere Ansprüche.

In Formel 4.4 ist ein Ratingfunktion, welche die genannten Punkte ausdrückt:

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

4.2.2 Dissensrating

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

Das Dissensrating ist eher auf einer Diskussionsplattformen denkbar. Dort können die Meinungen auseinander gehen. User*innen haben nicht unbedingt das Interesse die Position der Gegenüber einzunehmen.

Eine Funktion die die genannten Kriterien erfüllt verwendet die euklidische Distanz, es ergibt sich in Formel 4.5:

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

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, Veröffentlichungzeitpunkt
- 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 eine*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 $v : \mathbb{R}^n \to \mathbb{R}$.

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:

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

Die Differenz wird im Reddit Hot Ranking verwendet.

5.1.2 Anteil

Berechnet wird der Anteil der Upvotes zur Gesamtzahl der Votes:

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

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 abgegeben 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:

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

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 Gravitätskonstanste $G = 1.8$ potenziert. Die Bewertungstransformation lautet $v(b_P) = u_P - 1$ so ergibt sich die Hacker News Bewertungsmetrik in Formel 5.4:

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

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:

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

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.

$$e_P = r_P - E \tag{5.6}$$

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

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

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

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

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

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

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:

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

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:

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

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:

$$\tilde{s}_{\mu,p} = s_p + d_{\mu,p} \tag{5.12}$$

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:

$$\tilde{s}_{\sigma,P} = s_P + d_{\sigma,P} \tag{5.13}$$

6 Evaluationmethode

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.

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

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:

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

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:

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

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 vorgestelllten Iterationsevaluationen werden durch die Trapezregel in Formel 6.4 aggregiert und normiert:

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

6.2.2 Posts ohne Betrachtungen

Der Prozentsatz an Posts welche von keine*r einzig*en User*in betrachtet wurden werden durch die Funktion in Formel 6.5 gezählt:

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

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.jl[1] für Julia[2]. Die verwendete Version von Agents.jl ist ein weitereintwickelter Fork der Verion 3.0.0.

Sämtliche Funktionalität ist in dem Paket `VotingProtocols`[3] 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*innenschrittfunkion 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

[1] `agent.jl`
[2] `julialang.org`
[3] `github 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

```
model_configs = [
    (
        downvote_model,
        Dict(
            :scoring_function => scoring_reddit_hot,
            :model_step! => random_model_step!,
            :deviation_function => [
                mean_deviation,
                std_deviation],
        ),
    ),
    (
        [standard_model, downvote_model],
        Dict(
            :scoring_function => [
                scoring_activation,
                scoring_hacker_news],
            :gravity => [0.5, 1.0, 1.5, 2.0],
        ),
    ),
    (
        :all_models,
        Dict(
            :user_rating_function => user_rating_exp
        ),
    ),
]
```

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 Initalscore der Posts, die Bewertungsmetrik, die Relevanzgravität und die User*innenmeinungsfunktion ausgewertet.

```
\label{lis:eval}
iter_eval_functions =
[
  ndcg,
  gini,
  @gini_top_k(100)
]

model_eval_functions =
[
  posts_with_no_views,
  @area_under(:ndcg),
  @area_under(:gini),
  @area_under(:gini_top_100),
  @model_parameter(:init_score),
  @model_parameter(:scoring_function),
  @model_parameter(:relevance_gravity),
  @model_parameter(:user_rating_function),
]
```

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[4]. Die Datenanalyse kann so in R erfolgen.

[4] RFOTTNOT

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.

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

Voerst 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 ausgwertet. 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.

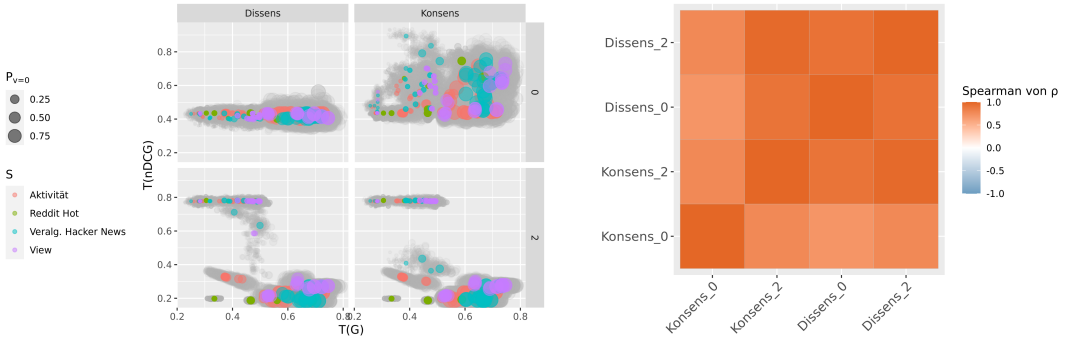

Figure 8.1: Fälle von Plattformen

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.

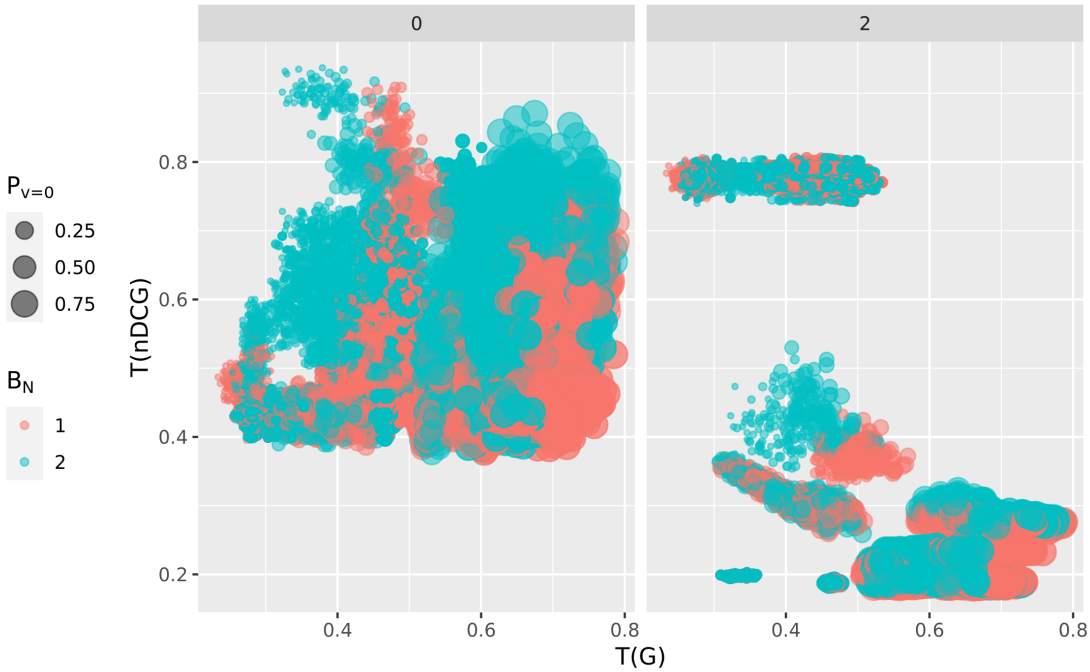

Figure 8.2: Bewertungsvektor

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 Initalwert. $s_0 = 0$ 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 $P_{v=0}$.

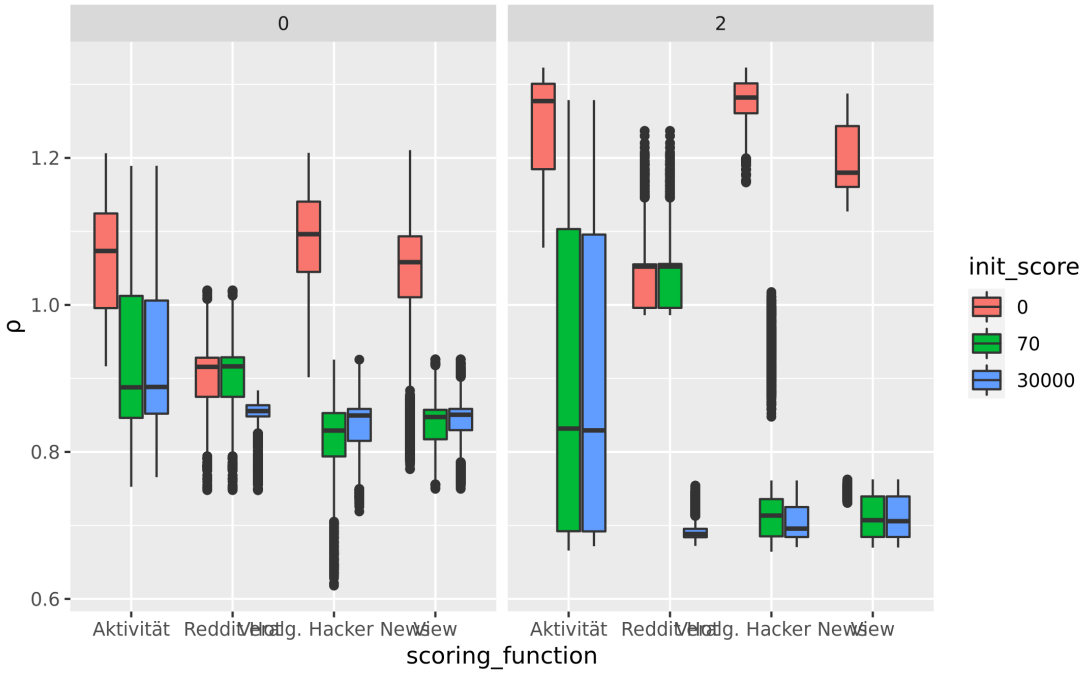

Figure 8.3: Initialscore

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

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

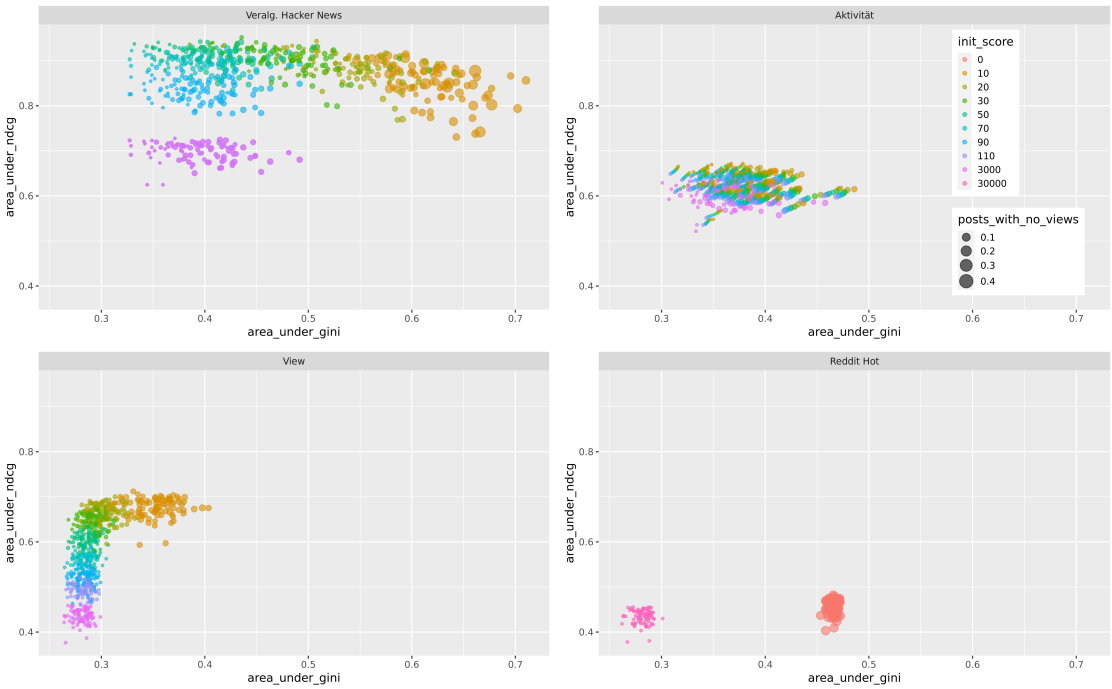

Figure 8.4: Initialscore in Bewertungsmetriken

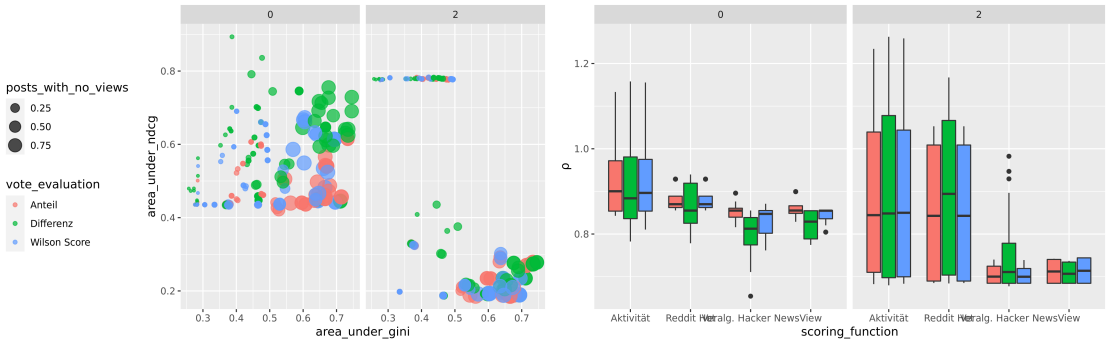

Figure 8.5: Bewertungstransformation

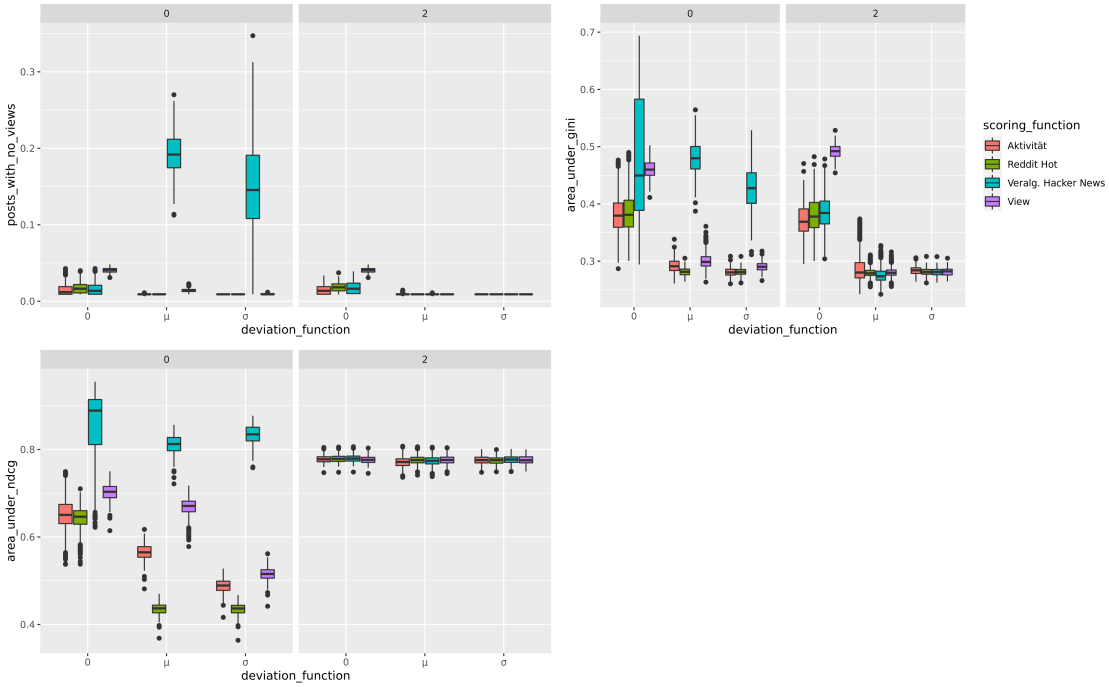

Figure 8.6: Zufallsbewertung

8.4 Zufallsbewertung

8.5 Bewertungsmetriken

8.6 Iterationslänge

8.7 User*innenparameter

8.7.1 User*innenwahrscheinlichkeitsfunktionen

8.7.2 User*innebewertungsfunktion

Dissensrating bringt erwartungsgemäß schlechtere Ergebnisse,

8.8 Bewertungsmetriken

8.8.1 Zufall

8.9 Modellparameter

8.9.1 Qualitätsraum

8.9.2 Vorsortierung

Verlauf kein Scatteplot ist bestimmt interessanter

8.9.3 (Extreme User*innen)

gleiche wie vorsorierteung mit verlauf

8.9.4 Relevanzgravity

Hacker News Derivate schneiden besser mit höherer Relevanz gravität ab
Nochmal mit höherer gravität versuchen 1.8 <
reddit auch für upvote model und hacker news auch für downvote models
zufälligkeit
Korrelation zwischen Ratingfunctions checken
Reihenfolge :
Initscore und gravität
Wenn gravität und relevanz gravität korrelieren kann diese zukunftig immer entsprechend der relevanzgravität gesetzt werden.
Modells mit init score testen vote evaluations funktionen
Korrelation zwischen Ratingfunctions checken default models, welche
Steps Start users und Posts
default models anlegen, welche den besten initscore vereinen und
Zwei Fälle relevance gravity = 0 und 2

9 Ausblick

- Sachen die noch berüchsigt werden könnten

Kommentare auf Posts

Im entwickelten Modell nimmt nur die Position im Rating sozialen Einfluss auf die Bewertungsentscheidung von Usern. Es wäre außerdem interessant den Einfluss zu untersuchen, welcher durch die Anzeige der aktuellen Bewertung des Posts entsteht

Dies wäre zu bewerkstelligen durch eine komplexere Kalkualtion der Bewertungswahrscheinlichkeit, in die dann nicht nur der skalare Wert des Users einfließt, sondern ebenfalls der betrachtete Post mit einfließt.

Dirichlet Smoothing

Konzentration besser modellieren und mit anderen User parametern korrelieren

Die Qualitätswahrnehmung von Usern ist im Modell nicht mit der Akitvität und der Bewertungswahrscheinlichkeit korreliert. Dies könnte jedoch in der Realität durchaus der Fall ist. Daher ist es vorstelllbar in einer Weiterentwicklung des Modells die Verteilungen zu korrelieren. Korrelationen von useraktivität, qauli

Eine einfache Umsetzung wäre denkbar, wenn die Dimensionsanzahl durch das Modell festgelegt ist

10 Conclusions

Formelzeichen

M	Modell
P	Post
U	User*in

Abkürzungsverzeichnis

M Model