Definition von Crowdsourcing-Kampagnen

Diplomarbeit

Richard Wotzlaw
Reitbahnstraße 36, 01069 Dresden

Dresden, den 15. Mai 2015

Technische Universität Dresden
Fakultät Informatik
Lehrstuhl Rechnernetze
Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill
Betreuer: Dipl.-Inf. Tenshi Hara, Dr.-Ing. Thomas Springer

Inhaltsverzeichnis

1 Einleitung

Mit dem Aufkommen des Internets und vor allem des World Wide Webs sind die Grenzen zwischen Medienkonsumenten und -produzenten immer weiter verschwommen. Insbesondere der Trend hin zu *User Generated Content*, der unter anderem dem *Web 2.0* genannten Phänomen ab etwa dem Jahr 2005 zugrunde lag, ließ das Web endgültig vom *Lean-Back-* zum *Lean-Forward-Medium* [Nie14] werden. Crowdsourcing ist die logische Schlussfolgerung, indem es auf die Beteiligung einer großen Menge an Nutzern setzt, deren Einzelbeiträge sich idealerweise gegenseitig z. B. zur Lösung einer Aufgabe oder zur Schaffung neuer Inhalte ergänzen. In dieser Arbeit soll der Hauptfokus auf Crowdsourcing-Kampagnen liegen, die eine Möglichkeit für einen Crowdsourcing-Anbieter sein sollen, genauere Angaben zur gewünschten Zielgruppe seines Vorhabens zu machen.

1.1 Motivation

Mit der Plattform SANE [HSBS13] wurde an der TU Dresden eine generische Crowdsourcing-Komponente entwickelt, deren Ziel es ist, eine Infrastruktur für beliebige Crowdsourcing-Anwendungen bereitzustellen. Bislang war es weitesgehend so, dass es am Crowdsourcer selbst war, diese Infrastruktur für sein gewünschtes Unterfangen aufzusetzen. Die Entwicklung von Apps, das Benutzermanagement, das Targeting geeigneter Teilnehmer und das Auswerten von Contributions sind nur einige der Aufgaben, die für ein gelungenes Crowdsourcing erledigt werden müssen.

SANE stellt eine zentrale Anlaufstelle für Crowdsourcer bereit, die ihnen einige dieser Arbeitsschritte abnimmt oder zumindest erleichtert. So können sich Crowdsourcer voll auf die Entwicklung von Apps für die Endanwender und die Auswertung von deren Beiträgen konzentrieren. Die Benutzerverwaltung wird vollständig von der SANE-Komponente erledigt und die Daten der Benutzer dabei immer nur anonymisiert an den Crowdsourcer weitergegeben. Dies bringt den weiteren Vorteil mit sich, dass ein Crowdsourcing-Teilnehmer nur dem SANE vertrauen muss und nicht immer neuen Crowdsourcing-Anbietern.

1.2 Zielsetzung

Der Fokus dieser Arbeit liegt auf der Definition von Crowdsourcing-Kampagnen, die es einem Crowdsourcer erlauben sollen, ihr Unterfangen genauer auf die gewünschte Zielgruppe auszurichten. Dies betrifft sowohl das Targeting geeigneter Teilnehmer als auch deren Belohnung. Idealerweise verbessert sich durch eine vom Crowdsourcer definierte Kampagne also sowohl die Motivation der Teilnehmer als auch die Qualität von deren Contributions.

Es soll dazu eine Definition von Crowdsourcing-Kampagnen aufgestellt werden, um genauer abzustecken, welche Anforderungen erfüllt werden müssen. Außerdem soll ein Format zur Definition einer Crowdsourcing-Kampagne in der Auszeichnungssprache XML entstehen und implementiert werden. Dazu gehört auch das Einbinden der ausgearbeiteten Konzepte in die bestehende Architektur der SANE-Software.

1.3 Struktur der Arbeit

Zunächst werden im Kapitel „Verwandte Arbeiten“ relevante Arbeiten anderer Autoren vorgestellt, die jeweils verschiedene Teilaspekte der angestrebten Eigenschaften von Crowdsourcing-Kampagnen beleuchten. Anschließend wird im Kapitel „Konzeption“ ein detailliertes Konzept zur Umsetzung solcher Kampagnen vorgestellt und ein Format zur Definition in XML entwickelt. Das Kapitel „Implementierung“ schildert dann die Umsetzung des Konzipierten und die Integration in die bestehende SANE-Architektur. Schließlich wird im Kapitel „Evaluation“ dargestellt, wie die implementierten Änderungen und ihre Erfolgsaussichten bewertet werden können.

Wann immer in dieser Arbeit Quelltextbeispiele verwendet werden, sind diese an der `nichtproportionalen Schrift` zu erkennen. Wann immer möglich und sinnvoll, wurde versucht, die im Text angesprochenen Teile der Konzeption oder Implementierung mit solchen Beispielen zu unterstützen. Ein Verzeichnis aller Quelltext-Listings findet sich, genauso wie auch das Literatur- und das Abbildungsverzeichnis, am Ende der Arbeit. Auch sind alle nichttrivialen Formeln und Gleichungen, die vor allem im Kapitel „Konzeption“ vorkommen, mit ausführlichen Berechnungsbeispielen erläutert worden.

2 Verwandte Arbeiten

Die faszinierenden Möglichkeiten, die Crowdsourcing bietet, in Kombination mit der scheinbaren Einfachheit des Konzepts haben in den letzten Jahren zu einem breiten Forschungsinteresse auf diesem Gebiet geführt. Nicht von Anfang hat war dabei der Begriff Crowdsourcing etabliert, frühe Veröffentlichungen sprachen beispielsweise von „peer production“ oder „user-powered systems“ [DRH11].

Neben Arbeiten, die sich mit der Definition von Crowdsourcing befassen, sollen an dieser Stelle auch Veröffentlichungen besprochen werden, die die in Bezug auf Crowdsourcing-Kamapgnen interessante Frage nach Anreizen zur Teilnahme untersuchen. Außerdem werden Arbeiten einbezogen, in denen Möglichkeiten zum Targeting bestimmter Teilnehmergruppen erörtert werden.

2.1 Der Begriff Crowdsourcing

Estellés-Arolas und González-Ladrón-de-Guevara führen in [EAGLDG12] eine Metaanalyse verschiedener Veröffentlichungen durch, um eine einheitliche und allumfassende Definition des Begriffs *Crowdsourcing* zu extrahieren. Sie identifizieren und aggregieren dazu wiederkehrende Elemente verschiedener Definitionen und Begriffsannäherungen und untersuchen darin Teilaspekte wie die Organisationsstruktur von Crowdsourcern und Teilnehmern (Unternehmen, NGOs, Einzelpersonen etc.), das Vorhandensein und die Art der Entlohnung der Teilnehmer und die Größe und Zusammensetzung der Crowd. Folgende Definition kristallisiert sich nach Meinung der Autoren heraus:

> *Crowdsourcing is a type of participative online activity in which an individual, an institution, a non-profit organization, or company proposes to a group of individuals of varying knowledge, heterogeneity, and number, via a flexible open call, the voluntary undertaking of a task. The undertaking of the task, of variable complexity and modularity, and in which the crowd should participate bringing their work, money, knowledge and/or experience, always entails mutual benefit. The user will receive the satisfaction of a given type of need, be it economic, social recognition, self-esteem, or the development of individual skills, while the crowdsourcer will obtain and utilize to their advantage that what* [sic] *the user has brought to the venture, whose form will depend on the type of activity undertaken.*
>
> — Estellés-Arolas und González-Ladrón-de-Guevara, 2012

Was in dieser Definition zu kurz kommt, sind Projekte, bei denen das kollektive Problemlösen lediglich ein Seiteneffekt der Benutzung eines Systems ist. Besonders hervorzuheben sind

hier Gamification-Ansätze, wie beispielsweise das in [Ahn06] vorgestellte Spiel *Peakaboom* oder das für iOS und Android erhältliche Spiel *Genes in Space*[1]. In beiden Fällen sind die Spiele derart gestaltet, dass die User (im Sinne des Crowdsourcings die *participants* oder *contributors*) durch die Tätigkeit des Spielens Daten generieren, die Wissenschaftlern (im Sinne des Crowdsourcings die *crowdsourcer*) bei der Beantwortung ihrer Fragen helfen, sei es durch Verifikation oder Falsifikation der Ergebnisse von Computer-Vision-Algorithmen oder die Analyse von Gensequenzen. Oftmals werden sich Benutzer dieser Anwendungen, wenn überhaupt, nur durch explizite Hinweise seitens der Crowdsourcer bewusst sein, dass sie an einem Crowdsourcing-Prozess teilnehmen.

Zwar findet dieser Aspekt auch bei Doan et. al. [DRH11] keine explizite Erwähnung, jedoch ist ihre Definition in dieser Hinsicht offener formuliert und nennt gleichzeitig zentrale Herausforderungen des Crowdsourcings:

> *We say that a system is a CS system if it enlists a crowd of humans to help solve a problem defined by the system owners, and if in doing so, it addresses the following four fundamental challenges: How to recruit and retain users? What contributions can users make? How to combine user contributions to solve the target problem? How to evaluate users and their contributions?*
>
> — Doan et. al., 2011

Im Besonderen für diese Arbeit interessant ist die letzte der genannten Herausforderungen: die Evaluation von Benutzern und deren Beiträgen. Aus der Evaluation wiederum Schlüsse zu ziehen, um von vornherein gezielt Teilnehmer mit den gewünschten Merkmalen auswählen zu können, ist eine notwendige Voraussetzung zur Definition einer Crowdsourcing-Kampagne.

2.2 Targeted Crowdsourcing

Crowdsourcing als Aktivität, die potentiell jeden Internetnutzer in den Prozess des Problemlösens einbeziehen kann, wirft natürlich die Frage nach der Eignung der Teilnehmer für die zu erledigende Aufgabe auf. Der Crowdsourcer möchte sicherstellen, dass das Problem von seiten der Teilnehmer möglichst effektiv und unter Einsatz möglichst geringer Kosten seinerseits gelöst werden kann. Einen Exploration-Exploitation-Algorithmus zur Auswahl des optimalen Workers pro Task haben Ho et. al. in [HJV13] vorgestellt, sich dabei aber auf Klassifizierungs-Aufgaben beschränkt.

2.2.1 Onlinewerbung

Ein potentiell universellerer Ansatz, der Internet-Werbung verwendet, wird von Ipeirotes und Gabrilovich in [IG14] mit dem System *Quizz* präsentiert. Die Aufgabe, die Nutzer zu bearbeiten haben, besteht dabei im Beantworten klassischer Quiz-Fragen zu einem bestimmten Fachgebiet. Die Qualität eines Nutzers – i. e. Informationen darüber, wie gut er im Quiz abgeschnitten hat – werden daraufhin dem Werbe-System (in diesem Fall Google

[1] `http://www.cancerresearchuk.org/support-us/play-to-cure-genes-in-space`, abgerufen am 11.11.2014

AdWords) als Feedback zugespielt, woraufhin dieses die Werbekampagne entsprechend optimieren kann.

Komponenten und Abläufe des Systems *Quizz*

Natürlich soll aber das Quiz den Teilnehmern nicht nur Fragen mit bekannten Antworten vorlegen, sondern die Kompetenz der User auch zur Wissensgewinnung verwenden. Zu diesem Zweck gibt es in *Quizz* neben den *calibration questions* eine zweite Art von Fragen: die *collection questions*, deren Beantwortung die eigentliche Contribution im Crowdsourcing-Prozess darstellt. Die wahrscheinliche Kompetenz des Users wird aus seiner Fähigkeit, die *calibration questions* zu beantworten, ermittelt, woraus sich dann ergibt, wie viel Vertrauen in seine Antworten auf die *collection questions* gesetzt werden kann. Einen Überblick über die Grundprinzipien von *Quizz* zeigt Abbildung 2.1.

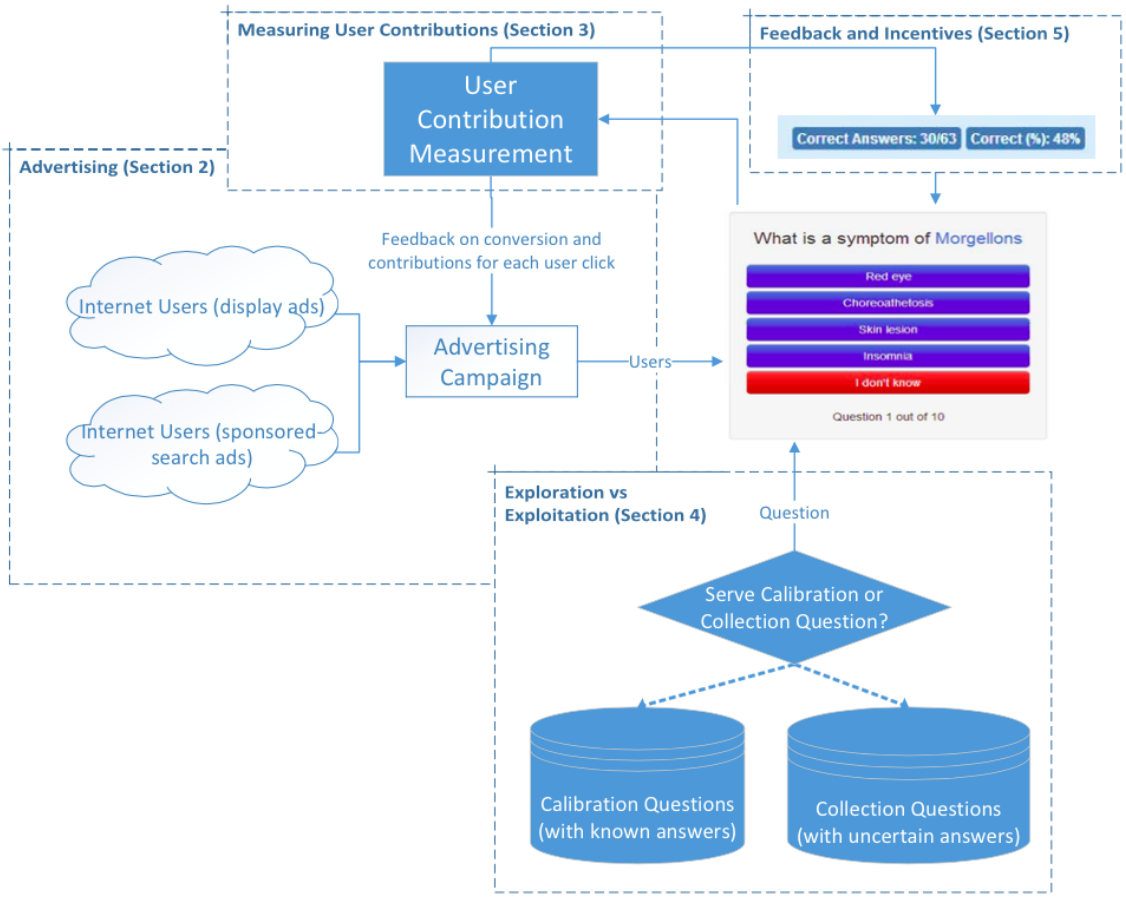

Abbildung 2.1: Überblick über das System *Quizz* [IG14]

Evaluation

Die Evaluation zeigt mehrere interessante Effekte. Zum einen korreliert die Anzahl der abgegebenen Antworten pro Teilnehmer mit der Güte dieser Antworten. Diese positive

Rückkopplung führt also langfristig dazu, dass schlechtere Teilnehmer schneller die Motivation verlieren und das Quiz beenden, was wiederum für das Crowdsourcing-Unterfangen vorteilhaft ist. Die Autoren führen weiterhin an, dass diese Beobachtung bei bezahltem Crowdsourcing, wie es beispielsweise bei Amazon Mechanical Turk vorliegt, in wesentlich schwächerer Ausprägung vorzufinden wäre, da Teilnehmer dort durch den monetären Anreiz über ihre Inkompetenz hinweggetröstet werden. Darüber hinaus hat sich die Rückkopplung der Teilnehmerqualität in den Advertising-Service als sehr effektive Maßnahme zur Steigerung der *conversion rate* pro Klick erwiesen:

> [...] *Thus, the cumulative difference was over 9.2x more answers obtained through the targeted campaign compared to the untargeted one (2866 answers vs. 279). Finally, the answers contributed by the users from the targeted campaign were of higher quality than the answers from the untargeted campaign: the total information gain for the targeted campaign was 11.4x higher than the total information gain for the untargeted one (7560 bits vs. 610 bits), indicating a higher user competence, even on a normalized, per-question basis.*
>
> — Ipeirotes und Gabrilovich, 2014

2.2.2 Einsatz von Diskriminatoren

Eine weitere Variante des Targetings beim Crowdsourcing stellen Li et. al. in [LZF14] vor. Grundsätzlich unterteilen sie den Crowdsourcing-Prozess in drei Phasen. Zunächst wird die Aufgabe in der *probing stage* jedem Interessierten vorgelegt, wobei gleichzeitig Informationen über die Teilnehmer in Form eines Fragebogens gesammelt werden. Darauf folgt die *discovery stage*, in der die Ergebnisse daraufhin analysiert werden, ob es unter den Teilnehmern anhand der durch die im Fragebogen gesammelten Diskriminatoren (auch *Features*) charakterisierbare Gruppen gibt, die signifikant besser abschneiden als andere. In der *targeting stage* schließlich wird der Rest der Aufgabe und zukünftige, ähnliche Aufgaben nur noch an die identifizierte Zielgruppe ausgespielt, wodurch in der Theorie mit weniger Teilnehmern das Problem ebenso effektiv gelöst werden kann.

Unterteilung der Crowd in Zielgruppen

Die Autoren führen zwei Möglichkeiten an, die vielversprechendste Zielgruppe für eine gegebene Aufgabe zu finden. Zum einen besteht die Grundidee des *Bottom-Up*-Ansatzes darin, die gesamte Crowd als Vielzahl kleiner Zielgruppen anzusehen, die durch die jeweils gesammelten Features der Teilnehmer entstanden sind. Nun können die Untergruppen, die performante Teilnehmer enthalten, zu der größeren Zielgruppe vereinigt werden. Features waren beispielsweise Bildungsstand, Studienfach und Geschlecht.

Der *Top-Down*-Ansatz dagegen betrachtet die Crowd als Ganzes und wählt dann nach und nach einzelne Features aus, um sie in Einzelgruppen zu unterteilen. Dabei wird jeweils die beste – i. e. performanteste – dieser Einzelgruppen zur weiteren Unterteilung gewählt. Schließlich würde man bei der bestmöglichen Zielgruppe ankommen. Abbildung 2.2 illustriert den Top-Down-Algorithmus: In einer Gruppe von 100 Teilnehmern mit den Features *Major*

und *Gender* stellt sich das Feature *Major* als signifikanter heraus. Innerhalb dieser Teilgruppe sind wiederum die *Science*-Studenten die performanteren. Im zweiten Schritt wird das Geschlecht betrachtet, da auch bei diesem Feature der Signifikanz-Schwellwert überschritten ist, wobei die Frauen besser abschneiden.

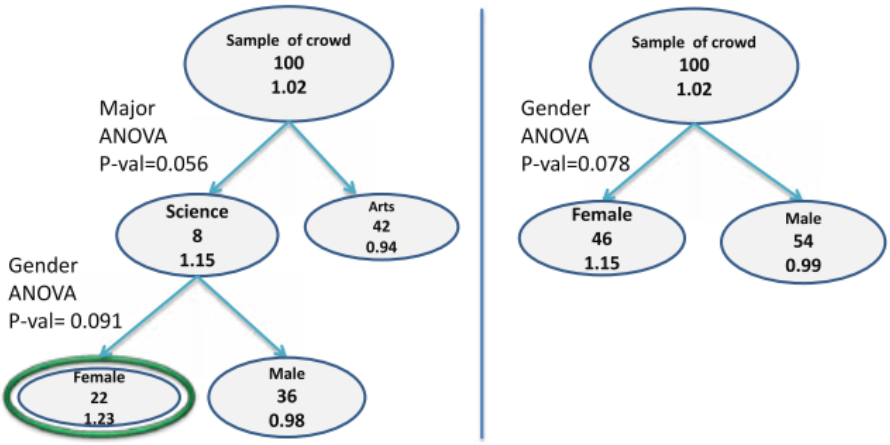

Abbildung 2.2: Fallbeispiel für den Top-Down-Algorithmus [LZF14]

Um eine zu kleinteilige Unterteilung in die Zielgruppen zu vermeiden, wird vorher ein *Feature Merging* durchgeführt. Dabei werden alle Ausprägungen eines Features – z. B. *Science*, *Engineering*, *Business* und *Arts* für das Feature *Major* – zu zwei Gruppen zusammengefasst. Ausschlaggebend ist dabei ein Ranking der Leistungen der einzelnen Teilnehmer jeder Teilgruppe.

Evaluation

Eine Auswertung wurde unter 100 Teilnehmern mit 75 Multiple-Choice-Fragen (mit bekannten Antworten) durchgeführt. Die ersten fünf Fragen waren dabei der *probing stage* zugeordnet. Da, wie bereits ausgeführt, vor dem Test die Features der Teilnehmer per Fragebogen abgefragt wurden, konnten mit diesen Daten die Zielgruppen sowohl im Sinne des Bottom-Up- als auch des Top-Down-Targetings bestimmt werden. Zwar gab es keine signifikanten Unterschiede zwischen den Targeting-Algorithmen, jedoch waren beide überlegen im Gegensatz zur ungezielten Verteilung der Fragen an zufällige Teilnehmer. Aus (leider unnötig selektiven) Diagrammen geht weiterhin hervor, dass das Feature *Major* (Studienhauptfach) vom Top-Down-Algorithmus über mehrere Durchläufe des Experiments mit Abstand am häufigsten als signifikant ausgewählt wurde. Als Ausprägungen dieses Features standen zur Auswahl *Science*, *Engineering*, *Literature*, *Arts* und *Other major*, von denen *Science* und *Other major* bei den Allgemeinwissensfragen am erfolgreichsten abschnitten. Zwei weitere Tests mit anderen Fragetypen brachten ähnliche Ergebnisse.

Fazit

Es wird deutlich, dass mithilfe von Diskriminatoren ein Targeting im Crowdsourcing möglich ist und die Ergebnisse unter Einsatz geringerer Kosten von gleicher oder besserer Qualität

sind. Allerdings ändern sich die optimalen Features, auf die das Targeting ausgelegt ist, je nach Art der Aufgabenstellung. Daher sollte das Format einer Kampagnendefinition in dieser Hinsicht möglichst flexibel sein und neben einer Festlegung auf bestimmte Charakteristika, die Teilnehmer zu erfüllen haben auch die dynamische Anpassung dieser Auswahlkriterien auf Basis der Qualität ihrer Contributions zulassen.

Auch der Einsatz eines Online-Werbesystems wie bei *Quizz* zeigt, dass systematische Optimierung des Targetings der vielversprechendste Weg hin zu hochwertigen Beiträgen geeigneter Teilnehmer ist. Beim *paid crowdsourcing* nach Art von Amazon Mechanical Turk wären derartige Effekte nach Meinung der Autoren wesentlich schwächer aufgetreten, da der Geldanreiz dort die intrinsischen Motivationsfaktoren der Teilnehmer überstrahlt hätte.

2.3 Teilnahmeanreize und -motivation

Eine wichtige Herausforderung für eine erfolgreiche Crowdsourcing-Kampagne ist die Motivation der Nutzer zur Teilnahme. Die Kampagne soll schließlich nicht nur optimale Nutzergruppen identifizieren, sondern auch viele und gute Contributions dieser Nutzer einsammeln können.

2.3.1 Allgemeine Motivationsfaktoren

Rotman et. al. untersuchen in [RPH+12] die Motivation von Crowdsourcing-Teilnehmern in Citizen-Science-Projekten[2]. Sie beziehen sich dabei auf die vier von Batson et. al. in [BAT02] identifizierten Formen der Motivation: *egoism*, *altruism*, *collectivism* und *principlism*.

> *Differentiation is based on identification of a unique ultimate goal for each motive. For egoism, the ultimate goal is to increase one's own welfare; for altruism, it is to increase the welfare of another individual or individuals; for collectivism, to increase the welfare of a group; and for principlism, to uphold one or more moral principles.*
>
> — Batson et. al., 2002

Für das initiale Interesse der freiwilligen Citizen Scientists zur Teilnahme fanden Rotman et. al. ausschließlich Motive, die sich dem *egoism* zuordnen lassen. So berichteten viele Teilnehmer der Studie, dass ihr persönliches Interesse am Thema des jeweiligen Projekts ein wichtiger Faktor sei und sie sich nicht vorstellen könnten, freiwillig an Projekten zu arbeiten, die an ihren persönlichen Interessengebieten vorbeigehen.

Anders verhielt es sich mit den Faktoren, die zur Aufrechterhaltung der Motivation bei den Teilnehmern führten. Hier stellte sich klar die Anerkennung ihrer wertvollen Beiträge als wichtigstes Element heraus:

[2] Wissenschaftliche Projekte, die via Crowdsourcing unter Mithilfe interessierter Laien bearbeitet werden. Beispiele reichen von der Erfassung von Lichtverschmutzung bis zum Zählen der Individuen einer Käferart.

> [...] *all volunteers emphasized the power of being recognized for their individual contributions as a crucial motivational factor. Although volunteers clearly accepted the distinct roles of scientists and volunteers in scientific work* [...]*, they nonetheless felt an intense need to be recognized and appreciated for their contributions, however big or small.*
>
> — Rotman et. al., 2012

Auch wenn Citizen Science nicht der zentrale Fokus dieser Arbeit ist, so findet sich die generelle Gegenüberstellung von Crowdsourcer und Teilnehmer bei allen Arten des Crowdsourcings, sei es in Form von Wissenschaftler und Laie oder Auftraggeber und Auftragnehmer.

2.3.2 Ausschüttung von Belohnungen

Harris [Har11] untersuchte den Einfluss verschiedener Anreizmodelle beim bezahlten Crowdsourcing. Er ließ die Teilnehmer auf der Plattform *Amazon Mechanical Turk* dabei anonymisierte Bewerbungsunterlagen für Manager-Posten auf einer Skala von 1 bis 5 bewerten. Als Goldstandard dienten die entsprechenden Bewertungen eines Personalers mit langjähriger Erfahrung. Die Teilnehmer wurden in vier gleich große Gruppen eingeteilt, von denen jede mit je einem der folgenden Anreizmodelle konfrontiert wurde:

- **kein Anreiz**: Die Teilnehmer bekamen \$0.06 pro bewerteter Bewerbung, ungeachtet der Qualität ihrer Bewertung verglichen mit dem Goldstandard.
- **positiver Anreiz**: Die Teilnehmer bekamen grundsätzlich \$0.06 pro bewerteter Bewerbung, konnten sich aber einen Bonus von weiteren \$0.06 pro Bewertung verdienen, falls diese mit dem Goldstandard übereinstimmte.
- **negativer Anreiz**: Die Teilnehmer bekamen grundsätzlich \$0.06 pro bewerteter Bewerbung, es sei denn ihre Bewertung unterschied sich von der des Goldstandards. In diesem Fall bekam der Teilnehmer für diese Bewertung nur \$0.03[3].
- **kombinierte Anreize**: Die Teilnehmer bekamen grundsätzlich \$0.06 pro bewerteter Bewerbung, konnten sich aber einen Bonus von weiteren \$0.06 pro Bewertung verdienen, falls diese mit dem Goldstandard übereinstimmte. Unterschied sich die Bewertung eines Teilnehmers von der des Goldstandards in mehr als der Hälfte der 48 bewerteten Bewerbungen, wurde seine Kompensation auf \$0.03 pro falscher Bewertung reduziert[3].

Die Auswertung zeigt einen signifikanten Qualitätsanstieg bei den drei Anreizmodellen im Gegensatz zur Variante ohne zusätzlichen Anreiz. Insbesondere lieferten die Modelle, die einen positiven Anreiz enthielten, verbesserte Ergebnisse. In Abbildung 2.3 ist der Zusammenhang zwischen den Anreizmodellen und der Qualität der Teilnehmerbewertungen dargestellt.

Außerdem zeigte sich, dass sich die Teilnehmer mit den Anreizen im Hinterkopf mehr Zeit für die Bearbeitung der Aufgabe ließen, wie in Abbildung 2.4 zu sehen.

[3] Die Reduktion ihrer Belohnung wurde den Teilnehmern vor Durchführung der Aufgabe nur angedroht, tatsächlich bekamen sie letztendlich doch den vollen Preis. Es ist zu vermuten, dass dieser Umstand auf technische Beschränkungen bei der Definition von HITs in Amazon Mechanical Turk zurückzuführen ist.

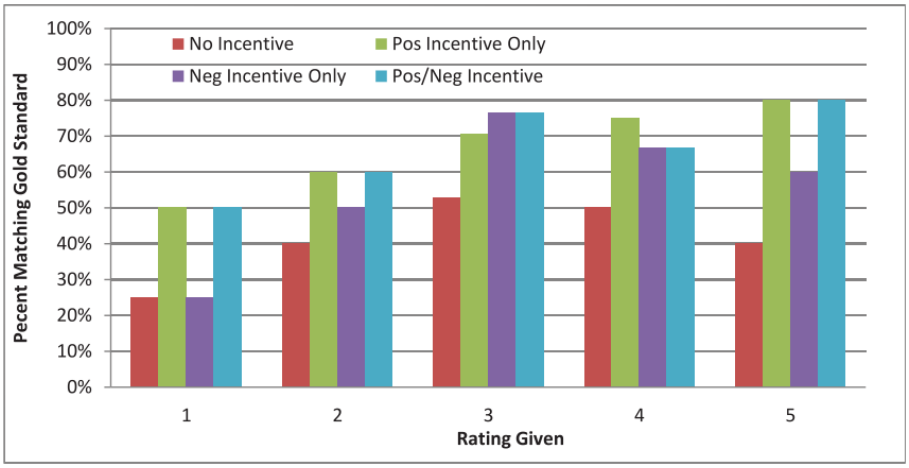

Abbildung 2.3: Anteil der mit dem Goldstandard übereinstimmenden Bewertungen der Teilnehmer in Abhängigkeit vom verwendeten Anreizmodell [Har11]

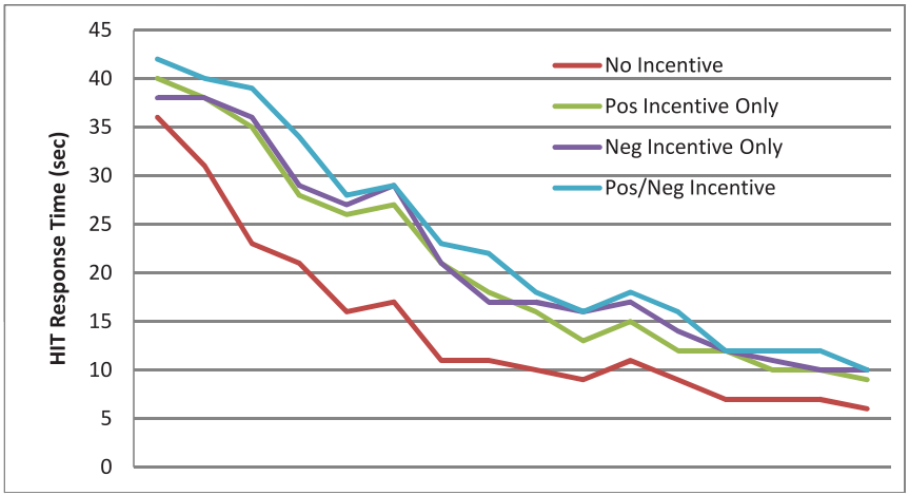

Abbildung 2.4: Zeit in Sekunden (y-Achse), die Teilnehmer im Durchschnitt für 16 exemplarische Bewerbungen (x-Achse) verbrauchten [Har11]

Ein zweiter, ähnlicher Versuchsaufbau wurde durchgeführt. Der Unterschied bestand darin, dass die Teilnehmer kein feingranulares Rating vornehmen, sondern lediglich Bewerbungen als *relevant* oder *nicht relevant* einstufen sollten. Auch in diesem Szenario gab es neben der Referenz-Aufgabenstellung mit fester Bezahlung pro Einstufung die drei Anreizmodelle *positiver Anreiz*, *negativer Anreiz* und *kombinierter Anreiz*, und erneut zeigte sich eine Leistungssteigerung bei den Teilnehmern, die mit Anreizen konfrontiert wurden, insbesondere mit positiven.

Arten von Belohnungen

Die naheliegendste Kompensation der Crowdsourcing-Teilnehmer ist Geld. Paid-Crowdsourcing-Plattformen wie *Amazon Mechanical Turk* zahlen den Bearbeitern der HITs[4] niedrige Centbeträge pro akzeptierter Contribution. Denkbar sind auch virtuelle Einheiten wie Punkte in einer Spielewelt. Abschnitt 2.3.3 beschäftigt sich näher mit derartigen Systemen.

Bei der Entwicklung von Open-Source-Software, die man bis zu einem gewissen Grad als Crowdsourcing betrachten kann, gibt es für die Entwickler (i. e. Crowdsourcing-Teilnehmer) keine materiellen Anreize oder Ausschüttung von Belohnungen. Die Motivation liegt eher in der Erweiterung der persönlichen Fähigkeiten, dem eigenen Bedürfnis nach der Benutzung der Software und der Anerkennung und möglicherweise damit verbundenem sozialen Aufstieg innerhalb der Entwicklergemeinschaft.

Einfluss der Höhe der Belohnung

Die Literatur zu den Effekten der Höhe der Belohnung ist relativ dürftig. Terwiesch und Xu [TX08] untersuchten verschiedene Belohnungsstrukturen in Crowdsourcing-Inititativen, legten den Fokus dabei aber auf den Bereich Open Innovation. Zum einen fanden sie Vorteile in Sachen Qualität der Contributions bei *Winner-Takes-It-All*-Modellen gegenüber Modellen mit Abstufungen in den Preisausschüttungen. Zum anderen zeigten sich leistungsabhängige Belohnungen als vorteilhaft gegenüber festgelegten Preisen. Fraglich ist dabei, inwieweit sich diese Ergebnisse auf allgemeine Crowdsourcing-Aufgaben übertragen lassen, die nicht unbedingt freie Ideenfindung erfordern.

Acar und van den Ende [AvdE11] postulieren zwar einen negativen Effekt der Größe der Belohnung auf die intrinsische Motivation der Teilnehmer, sehen dafür aber im Gegensatz eine Erhöhung der situativen extrinisischen Motivation mit steigender Belohnung. Eine Nutzerstudie, die ihre Theorien untersuchen würde, bieten die Autoren aber leider nicht an.

Fazit

Aussagen über die optimale Struktur und Höhe von Belohnungen im Crowdsourcing lassen sich aufgrund der spärlichen Literatur zu diesem Thema nicht treffen. Es ist aber davon auszugehen, dass extrinsische Motivation allgemein sowohl als initialer als auch als langfristiger Teilnahmeanreiz geeignet sein kann. Diese Belohnung muss nicht zwangsläufig in Form von Geld erfolgen. Stattdessen können virtuelle „Währungen“ (z. B. Punkte in einem Spielekontext) entwickelt werden, deren Erlangung innerhalb der Welt der Crowdsourcing-Anwendung den Teilnehmern Vorteile verschafft.

2.3.3 Gamification

Die Einführung spielerischer Elemente in kollaborativen Anwendungen hat sich in den vergangenen Jahren als ein beliebtes Instrument zur Förderung der Teilnehmermotivation

[4] Human Intelligence Tasks. Bezeichnung für die im Rahmen des Crowdsourcings von den Teilnehmern zu bearbeitenden Aufgaben bei Amazon Mechanical Turk

herausgestellt. Den Begriff „Gamification“, der sich dafür etabliert hat, definieren Deterding et. al. [DSN+11] wie folgt:

> *„Gamification“ is an informal umbrella term for the use of video game elements in non-gaming systems to improve user experience (UX) and user engagement.*
>
> — Deterding et. al., 2011

Beispiele für Crowdsourcing-Anwendungen mit Gamification-Elementen sind die bereits in Abschnitt 2.1 erwähnten Spiele *Peakaboom* und *Genes in Space*. Der Vorteil dieses Modells liegt auf der Hand: Beide beteiligten Seiten – sowohl Crowdsourcer als auch Teilnehmer – ziehen direkten Nutzen aus dem System, wodurch die Teilnahmemotivation gestärkt wird, was wiederum dem Crowdsourcer zugute kommt. Von Teilnehmern eigentlich als langweilig und eintönig empfundene Aufgaben wie Labeling von Bildern, Auswerten von Videoaufnahmen einer Überwachungskamera oder Korrektur von automatischen Textübersetzungen können durch spielerische Komponenten aufgelockert werden. Dadurch entstehende Konkurrenzsituationen zwischen Teilnehmern setzen plötzlich völlig neue Reize beim Bewältigen der Crowdsourcing-Aufgabe.

2.4 SANE

SANE (Server Access Network Entity) [HSBS13] ist eine an der TU Dresden entwickelte, generische Crowdsourcing-Infrastruktur. Auf Basis von SANE wird auch der in dieser Arbeit konzipierte Ansatz zur Definition von Crowdsourcing-Kampagnen umgesetzt. Die Idee hinter SANE ist es, „crowdsourcing as a service“ bereitzustellen. Ein solcher SANE agiert als Proxy zwischen Crowdsourcer und Teilnehmer. Zu seinen Aufgaben gehört das User-Management und die Anonymisierung der Benutzerdaten dem Crowdsourcer gegenüber. Dementsprechend werden die Contributions der Teilnehmer nur unter Angabe einer Submitter-ID an den Crowdsourcer weitergeleitet.

Mehrere SANE-Instanzen können gemeinsam als verteilte Crowdsourcing-Plattform agieren. Sie organisieren sich dabei selbst mithilfe einer Distributed Hash Table (DHT), wobei benachbarte Instanzen als gegenseitige Backups fungieren. SANE ist in der Programmiersprache PHP implementiert und verwendet MySQL zur Datenhaltung. Die Kommunikation mit Clients und Crowdsourcing-Backends wird über eine HTTP-Schnittstelle abgewickelt.

2.5 Zusammenfassung

Bezüglich des Targeting beim Crowdsourcing gibt es eine geringe Menge an Literatur, wobei unterschiedliche Ansätze verfolgt werden. Der Einsatz von Online-Werbung mit Google AdWords, das für eine automatische Optimierung der Kampagne sorgt, wie von Ho et. al. [IG14] vorgestellt, zeigt vielversprechende Ergebnisse und bestätigt die Annahme, dass geeignete Teilnehmerauswahl beim Crowdsourcing die Qualität der Contributions steigern kann. Ähnliche Erkenntnisse liefern Li et. al. [LZF14], die das Targeting mithilfe

von Diskriminatoren untersucht haben. Ihr Ansatz fokussierte sich auf zwei Algorithmen zur Bestimmung der geeignetsten Kriterien für eine gegebene Crowdsourcing-Aufgabe.

In der Arbeit von Harris [Har11] zeigten sich positive Effekte, wenn die Teilnehmer beim Crowdsourcing mit Erfolgsanreizen konfrontiert wurden. Solche Anreize können zum Beispiel eine Belohnung in Form von Geld sein, oder auch eine virtuelle Währung wie beispielsweise Belohnungspunkte innerhalb einer Spielumgebung. Dieser als *Gamification* bezeichnete Ansatz ist eine verbreitete Methode in vielen Crowdsourcing-Anwendungen.

3 Konzeption

Um eine geeignete Definition von Crowdsourcing-Kampagnen herauszuarbeiten, wird im Folgenden hergeleitet, was eine solche Kampagne ausmacht und welche Elemente sie enthalten muss. Danach wird auf die wesentlichen Elemente näher eingegangen und ein Format entwickelt, um Crowdsourcing-Kampagnen maschinenlesbar zu definieren.

3.1 Elemente von Crowdsourcing-Kampagnen

Eine Kampagne als „Phase, in der eine bestimmte Aktivität intensiv betrieben wird“ [ope14] muss definieren, welches Ziel diese Aktivität hat und wie es erreicht werden soll. Bei einer Crowdsourcing-Kampagne ist dieses Ziel klar: Die Qualität und/oder Quantität von Contributions soll gesteigert werden. Dazu muss ein Maß angegeben werden, mit dem die Qualität der Contributions gemessen werden kann. Außerdem soll die Angabe eines Targeting-Filters möglich sein. Dieser enthält Zielwerte für bestimmte Diskriminatoren, die Crowdsourcing-Teilnehmer erfüllen sollen, um hilfreich für das Erreichen des Kampagnenziels zu sein. Schließlich soll auch eine Belohnung der Teilnehmer erfolgen. Diese soll abhängig von der Eignung des Teilnehmers sein, um besonders wertvolle Nutzer weiterhin zur Teilnahme an der Crowdsourcing-Kampagne zu motivieren.

Um diesen Anforderungen gerecht zu werden, wird ein Konzept entwickelt, das die folgenden drei Elemente zur Definition einer Crowdsourcing-Kampagne vorsieht:

- **Target:** Wer soll zur Teilnahme an der Kampagne ausgewählt werden?
- **Ground Truth:** Wie soll die Qualität bzw. Eignung des Teilnehmers objektiv gemessen werden?
- **Reward:** Welche Belohnung erfolgt für hilfreiche Teilnehmer der Kampagne?

Auf jedes dieser drei Elemente wird im Folgenden näher eingegangen.

3.1.1 Target

Das Target ist ein Filter, der mithilfe von Diskriminatoren angibt, welche Teilnehmer für die Kampagne geeignet sind und welche nicht. In Abschnitt 3.2 wird genauer diskutiert, welche Diskriminatoren dafür sinnvoll sind. Die Teilnehmer können natürlich in unterschiedlichem Maße den Kriterien einer Kampagne genügen. Zu diesem Zweck kann in der Target-Definition für alle Diskriminatoren ein *Score* vergeben werden, den ein Teilnehmer zugewiesen bekommt, falls er das Kriterium erfüllt. Der Targeting-Gesamt-Score berechnet sich dann aus der Summe aller dieser Teilscores.

Arten von Verteilungen

Es soll grundsätzlich zwei verschiedene Varianten geben, den exakten Score pro Diskriminator zu berechnen. Zum einen per binärer Entscheidung: Trifft das Kriterium zu, bekommt der Teilnehmer/die Contribution für diesen Diskriminator den ausgelobten Score, sonst nicht. Die andere Möglichkeit ist eine graduelle Berechnung des Scores. Dazu sind in der Target-Definition zwei zusätzliche Angaben nötig: ein *Zentralwert* für den Diskriminator-Wert, der den *Maximal-Score* generieren soll und ein *Minimal-Score*, der genau für die Grenzwerte des Diskriminators generiert werden soll. Der jeweilige Score für den genauen Diskriminatorwert ermittelt sich dann mittels einer linearen Verteilung entlang dieser Werte.

Terminologie

Es werden für Target-Definitionen folgende Variablendeklarationen vereinbart:

- Sei s_{max} der maximal zu erreichende Target-Score.
- Sei s_{min} der Minimal-Score, der für Diskriminatorwerte an den Grenzen des Targets generiert werden soll.
- Sei $d_{zentral}$ der Zentralwert, also der Diskriminatorwert, der den Maximal-Score s_{max} generieren soll.
- Seien d_{min} und d_{max} der untere und obere Diskriminatorgrenzwert, für die jeweils der minimale Target-Score s_{min} generiert werden soll.
- Sei d der tatsächliche Diskriminatorwert.

Bei binären Score-Verteilungen sind $d_{zentral}$ und s_{min} nicht relevant. Hier gibt es nur einen zu erreichenden Score s_{max}, der ausgeschüttet wird, falls der Diskriminatorwert d innerhalb des in der Target-Definition angegebenen Intervalls $[d_{min}, d_{max}]$ liegt (bei numerisch darstellbaren Diskriminatoren) oder zu den gültigen Diskriminatorwerten gehört und für den konkreten Wert definiert ist (bei nicht numerisch darstellbaren Diskriminatoren).

Berechnung des Target-Scores für binäre Score-Verteilungen

Binäre Verteilungen können sowohl für numerische als auch für nichtnumerische Diskriminatoren angegeben werden, wobei sich die Target-Scores für ein Kriterium wie in den Gleichungen 3.1 und 3.2 angegeben berechnen. Die Zusammenhänge sind dabei trivial, weswegen an dieser Stelle auf eine Herleitung verzichtet wird.

Numerische Diskriminatoren. Ist im Target-Teil der Kampagnendefinition ein Diskriminator D mit einem Maximal-Score s_{max}, den Diskriminatorgrenzwerten d_{min} und d_{max} und einer binären Score-Verteilung definiert, berechnet sich der Score für einen Diskriminatorwert d aus Gleichung 3.1.

$$score_D(d) = \begin{cases} s_{max} & \text{wenn } d \in [d_{min}, d_{max}] \\ 0 & \text{wenn } d \notin [d_{min}, d_{max}] \end{cases} \tag{3.1}$$

Nichtnumerische Diskriminatoren. Falls der Diskriminator D nicht numerisch darstellbar ist und sich daher kein Intervall $[d_{min}, d_{max}]$ angeben lässt, soll die Definition stattdessen für jeden Diskriminatorwert einen eigenen Score s_d festlegen. Die Verteilung ist dann binär bezogen auf den Diskriminatorwert und nicht auf den Diskriminator. Gleichung 3.2 zeigt, wie sich der Score für den Diskriminatorwert daraus unmittelbar ergibt.

$$score_D(d) = s_d \tag{3.2}$$

Berechnung des Target-Scores für lineare Score-Verteilungen

Für den allgemeinen Fall eines Diskriminators D mit einer linearen Score-Verteilung, die den Maximal-Score s_{max} und den Minimal-Score s_{min} hat, den Diskriminatorgrenzwerten d_{min} und d_{max} und dem Zentralwert $d_{zentral}$ berechnet sich der Score für einen Diskriminatorwert d mithilfe der Funktion $score_D$ wie in Gleichung 3.3.

$$score_D(d) = \begin{cases} \dfrac{s_{max} - s_{min}}{d_{zentral} - d_{min}} \cdot (d - d_{min}) + s_{min} & \text{wenn } d \in [d_{min}, d_{zentral}[\\ \dfrac{s_{max} - s_{min}}{d_{zentral} - d_{max}} \cdot (d - d_{max}) + s_{min} & \text{wenn } d \in [d_{zentral}, d_{max}] \\ 0 & \text{wenn } d \notin [d_{min}, d_{max}] \end{cases} \tag{3.3}$$

Dabei wurde willkürlich für Fall 1 das halboffene Intervall angegeben, da sonst für $d = d_{zentral}$ sowohl Fall 1 als auch Fall 2 zuträfen. Letztendlich erzeugen ohnehin beide Fälle für $d = d_{zentral}$ denselben Score (nämlich den Maximal-Score s_{max}).

Herleitung. Die Herleitung dieses Zusammenhangs erfolgt mittels linearer Funktionen, die diese Verteilung erzeugen. Sie erfolgt an dieser Stelle für die „linke“ Seite des Verteilungsgraphen, also für die Berechnung der Scores für Diskriminatorwerte $d \in [d_{min}, d_{zentral}]$, ist aber völlig analog für die Berechnung der Scores für Diskriminatorwerte im Intervall $[d_{zentral}, d_{max}]$.

Um die linearen Funktionen $f(d) = md + n$ zu bestimmen, die zur Berechnung des Target-Scores für einen Diskriminatorwert d herangezogen werden können, wird zunächst der Anstieg m ermittelt. Von der linearen Funktion sind die beiden Punkte $P_1(d_{min}, s_{min})$ und $P_2(d_{zentral}, s_{max})$ bekannt. Damit ergibt sich der Anstieg m aus:

$$m = \frac{\Delta y}{\Delta x} = \frac{s_{max} - s_{min}}{d_{zentral} - d_{min}} \tag{3.4}$$

Einsetzen von m und P_1 in die allgemeine Funktionsgleichung und Umstellen nach n liefert:

$$n = s_{min} - \frac{s_{max} - s_{min}}{d_{zentral} - d_{min}} \cdot d_{min} \tag{3.5}$$

Damit ergibt sich folgende Funktionsgleichung zur Berechnung des Scores $f(d)$ in Abhängigkeit vom Diskriminatorwert d:

$$\begin{aligned} f(d) &= \frac{s_{max} - s_{min}}{d_{zentral} - d_{min}} \cdot d + s_{min} - \frac{s_{max} - s_{min}}{d_{zentral} - d_{min}} \cdot d_{min} \\ &= \frac{s_{max} - s_{min}}{d_{zentral} - d_{min}} (d - d_{min}) + s_{min} \end{aligned} \tag{3.6}$$

Analog ergibt sich die lineare Funktion $g(d)$ für Diskriminatorwerte im Intervall $[d_{zentral}, d_{max}]$ zu:

$$g(d) = \frac{s_{max} - s_{min}}{d_{zentral} - d_{max}} \cdot (d - d_{max}) + s_{min} \tag{3.7}$$

Zusammen mit dem Fall $d \notin [d_{min}, d_{max}]$ ergibt sich somit die abschnittsweise definierte Funktion aus Gleichung 3.3.

Berechnung des Target-Gesamt-Scores

Wie eingangs erwähnt, setzt sich der Score für das gesamte Target T aus der Summe aller Scores pro Diskriminator zusammen. Der Target-Gesamt-Score $score_T$ berechnet sich also aus einem geordneten Tupel von Diskriminatorwerten V und einem derselben Ordnung folgenden Tupel von Diskriminatoren X mit:

$$score_T(V) = \sum_{i=0}^{|X|-1} score_{X_i}(V_i) \tag{3.8}$$

Dabei bezeichnet X_i den Diskriminator an Position i im Tupel X und V_i den Diskriminatorwert an Position i im Tupel V. Die Nummerierung der Elemente eines Tupels beginnt hier per Vereinbarung bei 0.

Definitionsbeispiel

Als Beispiel soll eine Kampagnendefinition angenommen werden, die ein Target mit folgenden beiden Diskriminatoren angibt:

1. Das *Alter* der Teilnehmer soll zwischen 20 und 30 Jahren liegen und mit einem Score von 5 belegt sein. Der Score-Wert soll linear verteilt werden, sodass der maximale Score bei einem Alter von 25 Jahren liegt und der minimale Score 2 beträgt.
2. Der *Zeitpunkt* der Contribution soll zwischen 7 Uhr und 11 Uhr liegen und mit einem Score von 3 belegt sein. Der Score-Wert soll entweder vollständig oder gar nicht (binär) vergeben werden.

Berechnungsbeispiele

Wurde eine Target-Definition derart definiert, kann der Score eines Teilnehmers bzw. dessen Contribution exakt berechnet werden.

Zeitpunkt. Für das Kriterium *Zeitpunkt* ist dies trivial: Alle Contributions, die zwischen 7 Uhr und 11 Uhr (per Festlegung: inklusive) eingereicht werden, erhalten 3 Punkte, alle anderen 0. Die Funktion $score_{zeitpunkt}$ ist für eine Contribution zum Zeitpunkt d für dieses Kriterium also folgendermaßen definiert:

$$score_{zeitpunkt}(d) = \begin{cases} 3 & \text{wenn } 07{:}00{:}00 \leq d \leq 11{:}00{:}00 \\ 0 & \text{sonst} \end{cases} \tag{3.9}$$

Alter. Für das Kriterium *Alter* soll eine Linearverteilung des Scores angewendet werden. Auch hier gilt, dass allen Werten außerhalb der Grenzen (jünger als 20 oder älter als 30) ein Score von 0 zugewiesen werden soll. Liegt das Alter des Teilnehmers genau auf den Grenzen – i. e. ist der Teilnehmer genau 20 oder 30 Jahre alt – soll der Minimal-Score von 2 ausgeschüttet werden. Der Maximal-Score von 5 soll ausschließlich an 25-jährige Teilnehmer verteilt werden, da der angegebene Zentralwert bei 25 Jahren liegt. Zwischen diesen Altersangaben soll linear interpoliert werden. Es ergibt sich daraus eine Verteilung des Target-Scores wie in Abbildung 3.1 dargestellt.

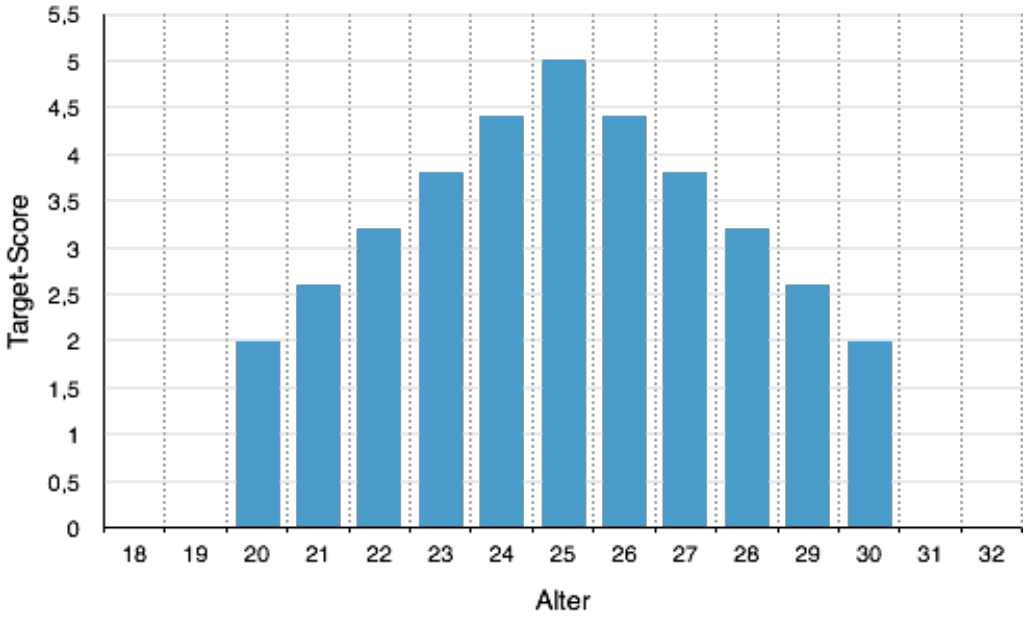

Abbildung 3.1: Verteilung des Target-Scores in Abhängigkeit vom Alter des Teilnehmers bei einem Zentralwert von 25, Maximal-Score von 5 und Minimal-Score von 2

Im Beispiel sind die Diskriminatorwerte für das *Alter* des Teilnehmers $d_{min} = 20$, $d_{zentral} = 25$ und $d_{max} = 30$. Für die erreichbaren Target-Scores wurden $s_{min} = 2$ und $s_{max} = 5$ festgelegt. Mithilfe der hergeleiteten Score-Funktion aus Gleichung 3.3 lässt sich nun die

Funktion $score_{alter}$ in Abhängigkeit vom tatsächlichen Alter des Teilnehmers d für dieses Kriterium wie in Gleichung 3.10 angeben.

$$score_{alter}(d) = \begin{cases} \frac{3}{5} \cdot d - 10 & \text{wenn } 20 \leq d < 25 \\ -\frac{3}{5} \cdot d + 20 & \text{wenn } 25 \leq d \leq 30 \\ 0 & \text{sonst} \end{cases} \tag{3.10}$$

Der vorgestellten Target-Definition folgend sollen nun beispielhaft die Target-Scores der Contributions zweier Teilnehmer berechnet werden. Die beiden Contributions erfüllen die beiden Target-Kriterien jeweils wie folgt:

1. Nutzer A ist 27 Jahre alt und gibt seine Contribution c_A um 08:34 ab.
2. Nutzer B ist 21 Jahre alt und gibt seine Contribution c_B um 11:10 ab.

Der Beitrag von Nutzer A wird innerhalb des im Target definierten Zeitfensters (zwischen 7 und 11 Uhr) abgegeben. Da der Diskriminator *Zeitpunkt* mit einem Score von 3 belegt ist und dieser binär (also ganz oder gar nicht) verteilt werden soll, bekommt Nutzer A für den *Zeitpunkt* einen Target-Score von $score_{zeitpunkt}(d_A) = 3$ zugewiesen (gemäß Gleichung 3.9).

Beim Diskriminator *Alter* soll eine lineare Verteilung des Target-Scores angewandt werden. Zur Berechnung des exakten Scores wird Gleichung 3.10 herangezogen. Da das Alter des Teilnehmers (27) im Interval $[d_{zentral}, d_{max}] = [25, 30]$ liegt, ist hier der zweite Fall anzuwenden. Der Target-Score für den Diskriminator *Alter* von Nutzer A berechnet sich also aus dem Diskriminatorwert $d_A = 27$ wie in Gleichung 3.11 angegeben.

$$score_{alter}(d_A) = -\frac{3}{5} \cdot d_A + 20 = -\frac{3}{5} \cdot 27 + 20 = 3{,}8 \tag{3.11}$$

Der Gesamt-Score von Teilnehmer A ist nun also die Summe dieser beiden Target-Scores und berechnet sich wie in Gleichung 3.12.

$$\begin{aligned} score_T(\text{'08:34'}, 27) &= score_{zeitpunkt}(\text{'08:34'}) + score_{alter}(27) \\ &= 3 + 3{,}8 = 6{,}8 \end{aligned} \tag{3.12}$$

Nutzer B gibt seine Contribution offensichtlich außerhalb des anvisierten Zeitfensters ab, bekommt für diesen Diskriminator also einen Target-Score von $score_{zeitpunkt}(\text{'11:10'}) = 0$. Gleichung 3.13 zeigt die Berechnung des Scores für den Diskriminator *Alter* mithilfe von Fall 1 aus Gleichung 3.10.

$$score_{alter}(d_B) = \frac{3}{5} \cdot d_B - 10 = \frac{3}{5} \cdot 21 - 10 = 2{,}6 \tag{3.13}$$

Damit ergibt sich der Gesamt-Score für die Contribution von Nutzer B wie in Gleichung 3.14.

$$\begin{aligned} score_T(\text{'11:10'}, 21) &= score_{zeitpunkt}(\text{'11:10'}) + score_{alter}(21) \\ &= 0 + 2{,}6 = 2{,}6 \end{aligned} \tag{3.14}$$

Da der Target-Score ein Maß dafür sein soll, wie gut ein Nutzer bzw. dessen Contribution für das Ziel der Crowdsourcing-Kampagne geeignet ist, würde die Kampagne in diesem Fall die Contribution von Nutzer A favorisieren, da deren Score größer ist.

Zentralwert = Mittelwert?

Beim Diskriminator *Alter* ergibt es sicherlich Sinn, als Zentralwert, also als das Alter, das den Maximal-Score generieren soll, genau den Mittelwert zwischen den Altersgrenzen zu wählen. Es könnte aber durchaus sinnvoll sein, etwa beim *Zeitpunkt* der Contribution einen Zentralwert festzulegen, der nicht genau in der Mitte der Grenzen liegt.

Angenommen, es sollen per Crowdsourcing in Echtzeit Börsenkurse prognostiziert werden. Der CEO eines Unternehmens tritt um 15 Uhr vor die Presse und stellt die neuesten Geschäftszahlen vor. Erst ab diesem Zeitpunkt sollen Contributions als wertvoll erachtet werden, allerdings auch nur für wenige Minuten. Das Zeitfenster ginge dann beispielsweise von 15:00 bis 15:05. Der Maximal-Score soll für die frühestmöglichen Contributions, also bereits um 15:00, ausgeschüttet werden. Dies wäre also der Zentralwert und gleichzeitig das Minimum der möglichen Diskriminatorwerte. Contributions um 15:05 sollen für diesen Diskriminator noch mit dem Minimal-Score belegt werden, spätere nur noch mit dem Score 0. Hier läge also eine „schiefe“ bzw. asymmetrische Verteilung der Scores vor, die wie in Abbildung 3.2 dargestellt aussehen könnte.

Nichtnumerische Diskriminatoren

Bisher wurde mit *Alter* und *Zeitpunkt* zwei Diskriminatoren besprochen, die numerisch darstellbar sind. In Abschnitt 3.2 werden sinnvolle Diskriminatoren für die Definition von Crowdsourcing-Kampagnen erläutert und dabei auch auch die Darstellbarkeit mithilfe von Score-Verteilungen eingegangen.

Für nicht numerisch erfassbare Kriterien scheiden lineare Score-Verteilungen dabei generell aus. Jedoch muss nicht immer pro Target-Kriterium ein fester Maximal-Score angegeben werden. Stattdessen können pro Diskriminatorwert Scores festgelegt werden, sodass beispielsweise der Diskriminator *Bildungsabschluss* für den Wert *Diplom* einen höheren Score generiert als für den Wert *Abitur*, letzteres aber immernoch einen höheren Score als *kein Abschluss*. In diesem Fall würde es sich ebenfalls um eine binäre Verteilung handeln, aber im Unterschied zu der bisher besprochenen Variante würden in diesem Beispiel den einzelnen Diskriminatorwerten jeweils eigene Scores zugewiesen, nicht dem Diskriminator als Ganzem.

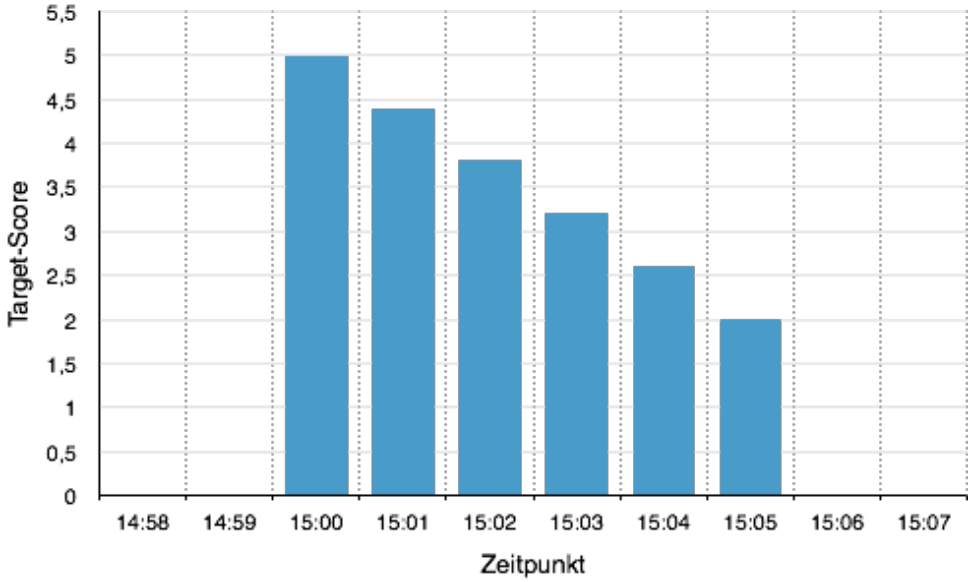

Abbildung 3.2: Verteilung des Target-Scores in Abhängigkeit vom Zeitpunkt der Contribution bei einem beispielhaften Zentralwert von $d =$ '15:00', den Diskriminatorgrenzwerten $d_{min} =$ '15:00' und $d_{max} =$ '15:05', dem Maximal-Score $s_{max} = 5$ und dem Minimal-Score $s_{min} = 2$

Score-Schwellwerte

Es soll weiterhin die Möglichkeit geben, auf der Target-Definition einen *Score-Threshold* anzugeben. Dieser soll einen Schwellwert darstellen, den der Target-Gesamt-Score erreichen muss, damit die Kampagne einen Teilnehmer bzw. eine Contribution als relevant ansieht. Dieser Schwellwert kann dann zum Beispiel benutzt werden, um über die Vergabe und ggf. die Höhe einer Belohnung des Teilnehmers (siehe Abschnitt 3.1.3) zu entscheiden.

3.1.2 Ground Truth

Die *Ground Truth* ist der Teil der Kampagnendefinition, mit dem sich die Qualität eines Crowdsourcing-Teilnehmers messen lässt. Es enthält eine Liste von Quizfragen mit bekannten Antworten, die dem Nutzer gestellt werden. Wiederum ist jeder Frage ein Score zugeordnet, der bei richtiger Beantwortung dem Teilnehmer zugewiesen wird. Der auf diese Weise erzielte Ground-Truth-Score kann zusammen mit dem Target-Score benutzt werden, um die Belohnung (siehe Abschnitt 3.1.3) eines Teilnehmers zu berechnen. Außerdem könnte das Wissen über die Expertise des Teilnehmers, das mithilfe der Ground-Truth-Fragen ermittelt wird, hilfreich sein für eine automatisierte Optimierung der Target-Definition (siehe Abschnitt 6.4).

Definition einer Ground-Truth-Frage

Eine Frage im Ground-Truth-Teil der Kampagnendefinition soll immer als *Multiple-Mark-Question* (MMQ) gestellt sein. Neben dem Fragentext enthält sie also eine beliebig lange

Liste vorgegebener Antwortmöglichkeiten, von denen eine beliebige Anzahl korrekt sein kann. Als Spezialfall sind somit auch klassische *Multiple-Choice-Questions* (MCQ) enthalten, die jeweils genau eine richtige Antwort haben. Außerdem ist, wie bereits erwähnt, jeder Frage ein Score zugeordnet; also eine maximale Anzahl an Punkten, die einem Teilnehmer bei korrekter Beantwortung zugewiesen werden und in seinen *Ground-Truth-Score* einfließen.

Ermittlung des Fragen-Scores

Die Natur von Multiple-Mark-Fragen wirft die Frage nach der exakten Berechnung des Scores auf.

Grundsätzliche Überlegungen. Naheliegenderweise kann nicht einfach verglichen werden, ob der Teilnehmer alle korrekten Antworten ausgewählt hat und ihm daraufhin die volle Punktzahl gegeben werden. Er könnte einfach alle Antworten gewählt haben, um auf jeden Fall alle richtigen Antworten abgedeckt zu haben.

Gleichzeitig ist es problematisch, ihn für korrekterweise nicht gewählte Antwortoptionen in gleichem Maße zu belohnen wie für korrekterweise gewählte. Dies würde dazu führen, dass ein Teilnehmer, der intuitiv betrachet „völlig falsch“ antwortet (z. B. weil er keine der korrekten Optionen gewählt hat, dafür aber mindestens eine der falschen Optionen tatsächlich nicht gewählt hat), trotzdem noch Punkte erhält. Insbesondere beim Spezialfall der MCQs leuchtet die Unsinnigkeit dieses Konzepts ein, denn dort würde niemand auf die Idee kommen, einem Teilnehmer trotz falscher Antwort noch Punkte dafür zu geben, zumindest nur eine der falschen Antworten gewählt zu haben (er konnte ja „by design“ nur eine Antwort wählen).

Ansatz von Tarasowa und Auer. Es gibt in der Literatur verschiedene Ansätze, sich dem Problem des *Scorings* bei MMQs zu nähern. Eine vielversprechende Berechnungsformel haben Tarasowa und Auer [TA13] vorgestellt. Sie beruht auf der Grundidee, zwischen der grundlegenden Bepunktung des Nutzers (den *basic points*, im Folgenden auch: Basispunkte) und der Strafe für anscheinendes Raten (der *penalty*, im Folgenden auch: Strafe) zu unterscheiden, und Letztere dann von Ersterem abzuziehen. Die Formel von Tarasowa und Auer soll auch in dieser Arbeit zur Berechnung des Ground-Truth-Scores verwendet werden.

Es werden zunächst folgende Variablendeklarationen festgelegt[1]:

- Sei s_{max} der für die aktuelle Frage maximal zu erreichende Score.
- Sei c_{max} die Anzahl der korrekten Antworten der aktuellen Frage.
- Sei $c_{gewählt}$ die Anzahl der vom Teilnehmer korrekterweise ausgewählten Antworten der aktuellen Frage.
- Sei a_{max} die Gesamtanzahl der Antwortmöglichkeiten der aktuellen Frage.
- Sei $a_{gewählt}$ die Anzahl der vom Teilnehmer insgesamt ausgewählten Antworten der aktuellen Frage.

[1] Die Variablen sind teilweise im Vergleich zur Originalarbeit umbenannt.

Nun können die Basispunkte s_{basic} einfach mittels des folgenden Zusammenhangs berechnet werden:

$$s_{basic} = \frac{s_{max}}{c_{max}} \cdot c_{gewählt} \tag{3.15}$$

Um die Strafe für das Raten eines Teilnehmers bei einer Frage zu ermitteln, weisen die Autoren sowohl der Frage selbst als auch den abgegebenen Antworten des Teilnehmers jeweils ein *level of guessing* zu, also Maße dafür, wie stark ein Teilnehmer vom Raten bei einer Frage theoretisch profitieren könnte und wie stark seine Antworten darauf hindeuten, dass er tatsächlich geraten hat. Das grundlegende *level of guessing* für eine Frage b und das *level of guessing* der Antworten des Teilnehmers n berechnen sich mithilfe der Gleichungen 3.16 und 3.17.

$$b = \frac{c_{max}}{a_{max}} \tag{3.16}$$

$$n = \frac{a_{gewählt}}{a_{max}} \tag{3.17}$$

Daraufhin ergibt sich die Strafpunktzahl $s_{penalty}$ des Teilnehmers für eine Frage wie in Gleichung 3.18 angegeben. Um zu verhindern, dass der Teilnehmer für das Nichtauswählen einer korrekten Antwort belohnt wird, indem ihm dies als Reduzierung des *user level of guessing* angerechnet wird und damit eine negative Strafpunktzahl generiert, darf $s_{penalty}$ nur für den Fall $n > b$ berechnet werden.

$$s_{penalty} = \begin{cases} (n-b) \cdot \dfrac{s_{max}}{1-b} & \text{wenn } n > b \\ 0 & \text{sonst} \end{cases} \tag{3.18}$$

Hieraus lässt sich nun die Funktion $score_q$ in Abhängigkeit vom Antwortmuster A eines Teilnehmers auf die Frage q angeben. Es ist zu beachten, dass die Punktzahl für eine Frage durch die *penalty* nicht negativ werden soll[2]:

$$score_q(A) = \begin{cases} s_{basic} - s_{penalty} & \text{wenn } s_{basic} > s_{penalty} \\ 0 & \text{sonst} \end{cases} \tag{3.19}$$

Berechnungsbeispiele. Als erstes Beispiel zur Berechnung des Ground-Truth-Scores soll eine klassische Multiple-Choice-Frage angenommen werden. Die Frage q_1 lautet „Wie viele Beine haben Spinnen?“ und die vorgegebenen Antwortmöglichkeiten sind: (a) vier, (b) sechs, (c) acht oder (d) zehn. Die einzig richtige Antwort ist (c) acht. Der Frage soll ein Score von $s_{max} = 4$ zugeordnet sein. Nutzer A beantwortet die Frage korrekt, wählt also Antwort

[2] Der Fall $s_{penalty} > s_{basic}$ wird in der Arbeit von Tarasowa und Auer nicht erwähnt. In dieser Arbeit wird aber die Festlegung getroffen, dass einem Teilnehmer keine negative Punktzahl für eine Ground-Truth-Frage zugewiesen werden soll.

(c) und sonst keine. Seine Antwort wird mit A_A bezeichnet und sein Score für diese Frage würde sich wie folgt berechnen:

$$\begin{aligned} score_{q_1}(A_A) &= \frac{s_{max}}{c_{max}} \cdot c_{gewählt} - (n-b) \cdot \frac{s_{max}}{1-b} \\ &= \frac{4}{1} \cdot 1 - \left(\frac{1}{4} - \frac{1}{4}\right) \cdot \frac{4}{1-\left(\frac{1}{4}\right)} \\ &= 4 \end{aligned} \tag{3.20}$$

Wie intuitiv zu erwarten war, bekommt der Teilnehmer hier für seine korrekte Antwort die volle Punktzahl. Hätte er falsch geantwortet (i. e. genau eine Antwort abgegeben, die aber falsch ist), wäre $c_{gewählt} = 0$ und damit auch $score_{q_1}(A_A) = 0$, was ebenfalls der Erwartung entspricht.

Wichtig hervorzuheben ist, dass die Berechnungsformel auch mit dem Fall umgehen kann, dass der Teilnehmer nicht von vornherein die Anzahl der korrekten Antworten kennt. Hätte ein Nutzer B beispielsweise neben der korrekten Antwort (c) zusätzlich noch Antwort (b) ausgewählt (weil ihm nicht gesagt wurde, dass nur eine Antwort richtig ist), hätte er zwar dieselben Basispunkte erhalten, dafür aber Strafpunkte abgezogen bekommen. Seine Punktzahl $score_{q_1}(A_B)$ hätte sich damit berechnet aus:

$$\begin{aligned} score_{q_1}(A_B) &= \frac{s_{max}}{c_{max}} \cdot c_{gewählt} - (n-b) \cdot \frac{s_{max}}{1-b} \\ &= \frac{4}{1} \cdot 1 - \left(\frac{2}{4} - \frac{1}{4}\right) \cdot \frac{4}{1-\left(\frac{1}{4}\right)} \\ &= 2{,}\overline{6} \end{aligned} \tag{3.21}$$

Nun soll die Berechnungsformel auch mit dem allgemeinen Fall von Multiple-Mark-Fragen umgehen können. Um dies zu zeigen, wird folgende MMQ q_2 angenommen: „Welche der folgenden Tennisspieler/-innen haben mindestens einmal das Einzel-Tennisturnier in Wimbledon gewonnen?“. Die Antwortoptionen sind (a) Kevin Curren, (b) Goran Ivanišević, (c) Andy Murray, (d) Ivan Lendl, (e) Stefan Edberg und (f) Arantxa Sánchez Vicario. Korrekt sind die Antworten (b), (c) und (e). Nutzer C beantwortet diese Frage und markiert die

Antwortmöglichkeiten (a), (c), (e) und (f). Vorausgesetzt $s_{max} = 9$, berechnet sich seine Punktzahl wie folgt:

$$\begin{aligned} score_{q_2}(A_C) &= \frac{s_{max}}{c_{max}} \cdot c_{gewählt} - (n-b) \cdot \frac{s_{max}}{1-b} \\ &= \frac{9}{3} \cdot 2 - \left(\frac{4}{6} - \frac{3}{6}\right) \cdot \frac{9}{1-\left(\frac{3}{6}\right)} \\ &= 3 \end{aligned} \tag{3.22}$$

Intuitiv sollte der Score steigen, wenn eine der falschen Antworten nicht abgegeben wird. Nutzer D beantwortet die Frage beispielsweise nur mit (c), (e) und (f). Tatsächlich erhält er damit, wie Gleichung 3.23 zeigt, eine höhere Punktzahl als Nutzer C, da Nutzer D keine Strafpunkte abgezogen bekommt.

$$\begin{aligned} score_{q_2}(A_D) &= \frac{s_{max}}{c_{max}} \cdot c_{gewählt} - (n-b) \cdot \frac{s_{max}}{1-b} \\ &= \frac{9}{3} \cdot 2 - \left(\frac{3}{6} - \frac{3}{6}\right) \cdot \frac{9}{1-\left(\frac{3}{6}\right)} \\ &= 6 \end{aligned} \tag{3.23}$$

Ermittlung des Gesamt-Scores

Ähnlich wie es bereits beim Target-Score der Fall war, ermittelt sich der Ground-Truth-Gesamt-Score aus der Summe aller Scores der einzelnen Fragen. Damit gilt für eine Ground-Truth-Definition G mit einer Liste an Fragen Q der Zusammenhang in Gleichung 3.24. A_{q_p} bezeichnet dabei jeweils das Antwortmuster von Teilnehmer p auf Frage q.

$$score_G(p) = \sum_{q \in Q} score_q(A_{q_p}) \tag{3.24}$$

Um für die Kampagne ungeeignete Teilnehmer ausschließen zu können, soll die Ground-Truth-Definition in der Kampagne – ähnlich wie bei der Target-Definition – die Möglichkeit für einen Score-Schwellwert vorsehen, den der Wert von $score_G$ mindestens erreichen muss.

Nachteile des Scoring-Verfahrens

Die Formel von Tarasowa und Auer berechnet für Fragen mit einem hohen Anteil korrekter Antworten und einem noch höheren Anteil vom Teilnehmer als korrekt markierter Antworten unter bestimmten Umständen kontraintuitive Punktzahlen. Ein Beispiel ist in Tabelle 3.1

dargestellt, in der für eine Frage q die Antwortmuster von zwei Teilnehmern bei einem Maximal-Score von $s_{max} = 12$ bewertet werden. Ein Häkchen ✓ steht dabei für eine korrekte (in der Spalte „Korrekt“) bzw. eine vom Teilnehmer als korrekt markierte Antwort. Im Fuß der Tabelle stehen die berechneten Punkte und in Klammern die Punkte, die bei vollständig korrekter Beantwortung theoretisch zu erzielen wären.

Antwort	**Korrekt**	**Teilnehmer A**	**Teilnehmer B**
(a)	✓	✓	✓
(b)		✓	✓
(c)	✓		
(d)		✓	✓
(e)	✓	✓	✓
(f)	✓	✓	
Basispunkte	(12)	9	6
Strafpunkte	(0)	6	0
Score	**(12)**	**3**	**6**

Tabelle 3.1: Vergleich des Scorings nach Tarasowa und Auer für die Antwortmuster der Teilnehmer A und B mit (scheinbar) widersprüchlichem Ergebnis

Obwohl Teilnehmer A mit Antwort (f) eine korrekte Antwort gewählt hat, die Teilnehmer B nicht gewählt hat (bei ansonsten gleichen Antworten), ist der Gesamt-Score von Teilnehmer A geringer. Der Grund ist die höhere *penalty*, da in deren Berechnung nur die Anzahl der abgegebenen Antworten eingeht. Es spielt dabei keine Rolle, ob der Teilnehmer richtige oder falsche Antworten markiert.

Schlussbemerkungen zu Ground-Truth-Fragen

Offensichtlich ist das Format der Fragen zur Ermittlung des Ground-Truth-Scores auf MMQs beschränkt. Dadurch ergeben sich Vorteile in der Einfachheit der Definition und der Auswertung. Vorsicht ist geboten bei Fragen, bei denen sich Antwortmöglichkeiten gegenseitig ausschließen oder bei denen eine oder mehrere Antwortmöglichkeiten in einer anderen subsummiert sind, z. B. bei Antwortoptionen wie „keine der genannten“ oder „alle Antworten sind richtig“. Die Punkteverteilung für solche Fragen würde mit der Formel von Tarasowa und Auer logischerweise verfälscht werden.

Ob den Teilnehmern die Anzahl der korrekten Antworten jeweils mitgeteilt werden sollte oder nicht, ist ein Implementierungsdetail und soll an dieser Stelle nicht weiter erörtert werden. Die Berechnungsformel von Tarasowa und Auer kann mit beiden Fällen umgehen, allerdings muss natürlich vom Crowdsourcer bei der Definition seiner Kampagne bedacht werden, dass die durchschnittliche Punktzahl der Teilnehmer bei bekannter Anzahl richtiger Antworten höchstwahrscheinlich steigen wird.

3.1.3 Reward

Im Reward-Teil der Kampagnendefinition soll angegeben werden können, wie ein Teilnehmer für eine Contribution belohnt werden soll. Dazu soll die Angabe einer Berechnungsformel möglich sein, in der zwei Variablen zur Verfügung stehen: der Target-Score und der Ground-Truth-Score (sofern es in der Kampagnendefinition diese Elemente gibt).

Die so berechneten Belohnungspunkte für eine Contribution eines Teilnehmers werden dann gespeichert und können von der Anwendung des Crowdsourcers in beliebiger Weise verwertet werden.

3.2 Diskriminatoren

Der Target-Teil einer Crowdsourcing-Kampagnendefinition soll auflisten, welche Kriterien geeignete Kampagnenteilnehmer erfüllen sollen. Dazu werden in diesem Abschnitt mögliche Diskriminatoren diskutiert und die dafür zu erfassenden Daten ermittelt. Abschließend soll auch die Verwendbarkeit für die in Abschnitt 3.1.1 vorgestellten Score-Verteilungen untersucht werden.

3.2.1 Diskussion verschiedener Diskriminatoren

Die Verwendungsmöglichkeiten verschiedener Diskriminatoren als Target-Kriterien einer Crowdsourcing-Kampagne werden in diesem Abschnitt untersucht und auf die jeweils zu beachtenden Besonderheiten eingegangen.

Alter

Das Geburtsdatum eines Nutzers ist leicht erfassbar und keine sonderlich sensible Information. Gleichzeitig kann die Angabe eines Alters oder Altersbereichs, in dem Teilnehmer liegen sollen, für den Crowdsourcer hilfreich sein, um die Kampagne gezielt auf Menschen mit einem bestimmten Erfahrungsschatz auszurichten. Sicher ist das Alter allein hierfür nur ein vages Indiz, jedoch ermöglicht es zumindest eine grobe Filterung gewünschter Teilnehmerkreise in dieser Hinsicht. Im einfachsten Fall kann es sich um eine Aussortierung minderjähriger Benutzer handeln, falls in der Crowdsourcing-Aufgabe beispielsweise nicht jugendfreie Inhalte vorkommen. Gerade in einem solchen Fall ist allerdings Vorsicht geboten, da es natürlich keine Garantie geben kann, dass das von der Anwendung erfasste Geburtsdatum der Wahrheit entspricht.

Das Alter ist offensichtlich ein numerischer Diskriminator, demzufolge sind sowohl binäre als auch lineare Score-Verteilungen möglich. Zur Datenhaltung ist zu einem Crowdsourcing-Teilnehmer sein Geburtsdatum zu erfassen. Mit dem genauen Geburtsdatum kann das Alter bis auf den Tag genau ausgewertet werden, was noch feinere Score-Verteilungen ermöglicht, als würde man lediglich das Alter als (ganze) Anzahl an Jahren betrachten.

Zeitpunkt

In zeitkritischen Crowdsourcing-Initiativen kann der Zeitpunkt einer Contribution sehr entscheidend sein. Soll das Verkehrsaufkommen auf vielbefahrenen Straßen per Crowdsourcing ermittelt werden, sind Contributions zur Rush Hour wertvoller als Einsendungen um 2 Uhr nachts. Außerdem ist für den Zeitpunkt überhaupt keine weitere Datenerfassung nötig, da die Anwendung bzw. der Crowdsourcing-Server diese Information einfach direkt ermitteln und zusammen mit der Contribution selbst ablegen kann. Der Zeitpunkt ist ebenfalls ein numerischer Diskriminator, erlaubt also binäre und lineare Score-Verteilungen. In der Kampagnendefinition soll es möglich sein, sowohl einen Zeitraum an einem konkreten Datum als Target-Kriterium anzugeben, als auch eine wiederkehrende Zeitspanne (wie z. B. „jeden Tag zur Rush Hour von 16 bis 19 Uhr“).

Ort

Die Erfassung der Location eines Teilnehmers zum Zeitpunkt der Contribution eröffnet eine Vielzahl von Anwendungsmöglichkeiten für Crowdsourcing.

Allgemeine Betrachtungen. Unglücklicherweise steht das Auswerten dieser Information im Widerspruch zum Privatsphärebedürfnis vieler Benutzer. Aus diesem Grund sollen die Geokoordinaten, die ein Teilnehmer sendet, nur obfuskiert gespeichert werden. Dazu soll die Location des Teilnehmers innerhalb eines Kreises mit festgelegtem Radius m verschoben werden. Auf diese Weise kann der Ort weiterhin als ein Punkt abgelegt werden, von dem nur bekannt ist, dass der wahre Ort des Nutzers sich in einem Umkreis um diesen Punkt mit Radius m befindet[3].

Es sind dann binäre und lineare Score-Verteilungen möglich, wobei bei einer linearen Verteilung eine Maximaldistanz von der Target-Location angegeben werden muss, für die noch der Minimal-Score s_{min} vergeben wird. Der eigentliche Diskriminatorwert ist dann die Entfernung von der Target-Location und trivialerweise gilt $d_{min} = 0$. Die angegebene Maximaldistanz muss mindestens das Doppelte des Obfuskationsradius m betragen, um den Fehler aufgrund der Verschleierung der Location abzufangen.

Location-Verschleierung. Ist eine vom Teilnehmer gesendete Location Loc_{real} mit den Koordinaten (lat_{real}, lon_{real}) gegeben, soll diese obfuskiert, das heißt innerhalb eines Kreises mit dem Radius m verschoben, als Location Loc_{obf} mit den Koordinaten (lat_{obf}, lon_{obf}) gespeichert werden. Bei den folgenden Berechnungen wird die Erde als ideale Kugel mit dem Erdradius $r = 6378{,}137\,\text{km}$ betrachtet. Der Algorithmus zur Verschiebung der Teilnehmer-Location arbeitet wie folgt:

1. Berechne die maximale Verschiebung Δlat_{max} des Breitengrads lat_{real}, die noch innerhalb eines Kreises mit Radius m fallen würde.

[3] Ungenauigkeiten in der Positionsbestimmung per GPS treten natürlich zusätzlich auf.

Der Abstand der Breitengrade im idealen Kugelmodell der Erde beträgt konstant $d_{lat} = 111{,}319\,49\,\text{km}$. Daher gilt für Δlat_{max} folgende Gleichung:

$$\Delta lat_{max} = \frac{m}{d_{lat}} \tag{3.25}$$

2. Wähle eine zufällige Verschiebung Δlat für den Breitengrad lat_{real} mit $0 \leq \Delta lat \leq \Delta lat_{max}$.

3. Entscheide zufällig, ob $lat_{real} \geq lat_{obf}$ oder $lat_{real} \leq lat_{obf}$ gelten soll (Verschiebung in Nord- oder Südrichtung).

4. Berechne lat_{obf} gemäß der Entscheidung in Schritt 3, also entweder aus Gleichung 3.26 oder aus Gleichung 3.27, und konvertiere bei eventuellem Überlauf ins Intervall $[-90°, 90°]$.

$$lat_{obf} = lat_{real} + \Delta lat \tag{3.26}$$

$$lat_{obf} = lat_{real} - \Delta lat \tag{3.27}$$

5. Berechne die maximale Verschiebung Δlon_{max} des Längengrads lon_{real}, die zusammen mit der Verschiebung des Breitengrads Δlat für eine Gesamtverschiebung von Loc_{real} um m sorgen würde.

   Dazu wird das Dreieck auf der Erdoberfläche, das durch den Nordpol, Loc_{real} und Loc_{obf} aufgespannt wird, betrachtet. Darin wird mithilfe des Seitenkosinussatzes der sphärischen Trigonometrie folgender Zusammenhang (für Winkel im Bogenmaß) hergeleitet:

$$\begin{aligned}\cos\left(\frac{m}{r}\right) &= \cos\left(\frac{\pi}{2} - lat_{real}\right) \cdot \cos\left(\frac{\pi}{2} - lat_{obf}\right) \\ &\quad + \sin\left(\frac{\pi}{2} - lat_{real}\right) \cdot \sin\left(\frac{\pi}{2} - lat_{obf}\right) \\ &\quad \cdot \cos\left(lon_{real} - lon_{obf}\right) \\ &= \sin\left(lat_{real}\right) \cdot \sin\left(lat_{obf}\right) + \cos\left(lat_{real}\right) \cdot \cos\left(lat_{obf}\right) \\ &\quad \cdot \cos\left(\Delta lon_{max}\right)\end{aligned} \tag{3.28}$$

   Daraus ergibt sich für Δlon_{max} die Gleichung:

$$\Delta lon_{max} = \arccos\left(\frac{\cos\left(\frac{m}{r}\right) - \sin\left(lat_{real}\right) \cdot \sin\left(lat_{obf}\right)}{\cos\left(lat_{real}\right) \cdot \cos\left(lat_{obf}\right)}\right) \tag{3.29}$$

6. Analog zu Schritt 2: Wähle eine zufällige Verschiebung Δlon für den Breitengrad lon_{real} mit $0 \leq \Delta lon \leq \Delta lon_{max}$.

7. Analog zu Schritt 3: Entscheide zufällig, ob $lon_{real} \geq lon_{obf}$ oder $lon_{real} \leq lon_{obf}$ gelten soll (Verschiebung in Ost- oder Westrichtung).

8. Analog zu Schritt 4: Berechne lon_{obf} gemäß der Entscheidung in Schritt 7, also entweder aus Gleichung 3.30 oder aus Gleichung 3.31, und konvertiere bei eventuellem Überlauf ins Intervall $[-180°, 180°]$.

$$lon_{obf} = lon_{real} + \Delta lon \tag{3.30}$$

$$lon_{obf} = lon_{real} - \Delta lon \tag{3.31}$$

9. Speichere die so gewonnene, obfuskierte Location $Loc_{obf}(lat_{obf}, lon_{obf})$ als Diskriminatorwert für die Location des Teilnehmers bzw. seiner Contribution ab.

Berechnungsbeispiel. Gegeben sei die vom Teilnehmer gesendete Location Loc_{real} mit den Koordinaten $(lat_{real}, lon_{real}) = (43{,}314\,252° | -3{,}009\,216°)$. Diese soll verschleiert werden, indem sie innerhalb eines Kreises mit Radius $m = 50\,\text{km}$ verschoben wird. Der soeben vorgestellte Algorithmus berechnet den obfuskierten Geopunkt wie folgt:

1. Mit $m = 50\,\text{km}$ und $d_{lat} = 6378{,}137\,\text{km}$ ergibt sich Δlat_{max} zu:

$$\begin{aligned} \Delta lat_{max} &= \frac{50\,\text{km}}{6378{,}137\,\text{km}} \approx 7{,}839\,279\,714 \cdot 10^{-3}\,\text{rad} \\ &\approx 0{,}449\,157\,642° \end{aligned} \tag{3.32}$$

2. Es wird zufällig eine Verschiebung von $\Delta lat = 0{,}3°$ gewählt.

3. Es wird zufällig festgelegt, dass $lat_{real} > lat_{obf}$ sein soll, also der Ausgangspunkt nach Süden verschoben wird.

4. lat_{obf} wird gemäß Gleichung 3.27 bestimmt:

$$lat_{obf} = 43{,}314\,252° - 0{,}3° = 43{,}014\,252° \tag{3.33}$$

5. Zunächst werden lat_{real} und lat_{obf} ins Bogenmaß umgerechnet:

$$lat_{real} = 43{,}314\,252° \approx 0{,}755\,976\,310\,\text{rad} \tag{3.34}$$

$$lat_{obf} = 43{,}014\,252° \approx 0{,}750\,740\,322\,\text{rad} \tag{3.35}$$

Anschließend wird Gleichung 3.29 angewandt, um Δlon_{max} zu bestimmen:

$$\Delta lon_{max} \approx 7{,}998\,819\,859 \cdot 10^{-3}\,\text{rad} \approx 0{,}458\,298\,619° \tag{3.36}$$

6. Es wird zufällig eine Verschiebung von $\Delta lon = 0{,}2°$ gewählt.
7. Es wird zufällig festgelegt, dass $lon_{real} > lon_{obf}$ sein soll, also der Ausgangspunkt nach Westen verschoben wird.
8. lon_{obf} wird gemäß Gleichung 3.31 bestimmt:

$$lon_{obf} = -3{,}009\,216° - 0{,}2° = -3{,}209\,216° \tag{3.37}$$

9. $(lat_{obf}, lon_{obf}) = (43{,}014\,252°|-3{,}209\,216°)$ sind die Koordinaten der obfuskierten Teilnehmer-Location Loc_{obf}.

Eine nach diesem Algorithmus berechnete Location liegt garantiert innerhalb eines Umkreises mit dem Radius m (in diesem Fall 50 km) um Loc_{real}. Im Beispiel beträgt die Entfernung etwa 37,13 km. Abbildung 3.3 zeigt die im Beispiel verwendete, reale Location Loc_{real} und die daraus berechnete obfuskierte Location Loc_{obf}.

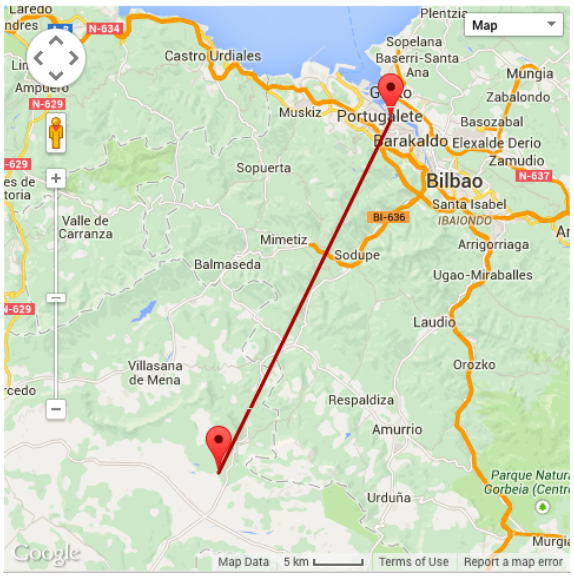

Abbildung 3.3: Aus der tatsächlich vom Teilnehmer gesendeten Location Loc_{real} (oben rechts) berechnete, obfuskierte Location Loc_{obf} (unten links) [via Google Maps]

Mögliche Erweiterungen. Eine möglicherweise sinnvolle Erweiterung der Obfuskierung würde darin bestehen, den Wert des Verschiebungsradius m veränderlich zu gestalten. Naheliegend wäre eine Verkleinerung von m in Ballungsgebieten und großen Städten und eine Vergrößerung in ländlichen Gegenden, da letztendlich die Bevölkerungszahl innerhalb des Verschleierungsradius ausschlaggebend für die Effektivität der Verschleierung ist. Diese Verbesserung des Obfuskierungs-Algorithmus wird in dieser Arbeit nicht weiter verfolgt, kann aber für zukünftige Erweiterungen in Betracht gezogen werden.

Bildung/Ausbildung

Da es bei vielen Crowdsourcing-Anwendungen um die Ausnutzung der Kenntnisse und Fähigkeiten der Teilnehmer geht, könnte der Bildungsstand ein geeigneter Indikator sein. Natürlich kann die (Aus)bildung eines Crowdsourcing-Teilnehmers unmöglich vollumfänglich erfasst werden. Für spezielle Kategorien wie Sprachkenntnisse wurden eigene Diskriminatoren konzipiert, ansonsten bleibt nur die Möglichkeit, den höchsten Bildungsabschluss des Teilnehmers zu erfassen, wobei eine gesonderte Untersuchung über die Aussagekraft dieses Kriteriums für das Crowdsourcing durchgeführt werden müsste. Mögliche Diskriminatorwerte sind:

- kein Abschluss
- Hauptschulabschluss
- mittlere Reife
- Abitur
- Berufsausbildung
- Vordiplom/Bachelor
- Diplom/Master
- Promotion
- Habilitation

Die Score-Verteilung erfolgt binär, wobei jedem Diskriminatorwert ein eigener Score zugewiesen werden kann.

Sprachkenntnisse

Viele Crowdsourcing-Aufgaben liegen im Bereich der Textübersetzung bzw. Übersetzungskorrektur, da Algorithmen in diesen Bereichen nicht ausgereift genug sind, um derartige Tätigkeiten allein rechnergestützt durchzuführen. Folglich ist es wünschenswert, in einer Kampagne auf Sprecher einer bestimmten Sprache abzuzielen. Neben der Liste an Sprachen, die ein Nutzer beherrscht, sollte dabei auch das Sprachniveau mit erfasst werden. So möchten Crowdsourcer für Übersetzungen *in* eine Zielsprache wahrscheinlich Muttersprachler dieser Sprache adressieren. Für Übersetzungen *aus* einer Sprache ist es hingegen weniger wichtig, ob diese Sprache die Muttersprache des Teilnehmers ist.

Es sollen also von den Teilnehmern die Sprachen und das jeweilige Sprachniveau abgefragt werden. Dies kann evtl. durch Ground-Truth-Fragen unterstützt werden, um diese möglicherweise recht unverlässliche Angabe zu verifizieren. Die einzig mögliche Score-Verteilung beim Diskriminator *Sprache(n)* ist eine binäre, wobei pro Sprache ein eigener Score definiert werden soll.

Belohnungspunkte

Die bisher für das erfolgreiche Bearbeiten von Crowdsourcing-Aufgaben erzielten Belohnungspunkte (siehe Abschnitt 3.1.3) sind ein wertvoller Indikator für die Qualität der Contributions eines Teilnehmers. Besonders hilfreich kann die Möglichkeit sein, sehr unerfahrene Nutzer anhand der geringen Zahl ihrer Contributions direkt auszuschließen. Außerdem entsteht eine positive Rückkopplung, wenn Teilnehmer Belohnungspunkte durch das Bearbeiten einer Crwodsourcing-Aufgabe sammeln, wodurch sich wiederum ihr Target-Score einer Kampagne steigert, was dann wiederum zu einer höheren Belohnung führt usw. Es sind sowohl binäre als auch lineare Score-Verteilungen möglich.

3.2.2 Überblick

Tabelle 3.2 zeigt alle diskutierten Diskriminatoren im Überblick.

Diskriminator	**Erforderliche Datenerfassung**	**Score-Verteilungen**
Alter	Geburtsdatum des Teilnehmers	binär, linear
Zeitpunkt	Timestamp der Contribution	binär, linear
Ort	obfuskierte Location des Teilnehmers	binär, linear
Bildung/Ausbildung	höchster Bildungsabschluss des Teilnehmers	binär
Sprachkenntnisse	Liste der Sprachen des Teilnehmers und jeweiliges Sprachniveau	binär
Belohnungspunkte	Belohnungspunkte pro Contribution	binär, linear

Tabelle 3.2: Zur Definition einer Crowdsourcing-Kampagne zur Verfügung stehende Diskriminatoren, ihre benötigten Daten und mögliche Score-Verteilungen

3.3 Definition von Crowdsourcing-Kampagnen

Ausgehend von den bisher konzipierten Eigenschaften von Crowdsourcing-Kampagnen sollen in diesem Abschnitt konkrete Möglichkeiten zur Spezifikation einer Kampagne durch einen Crowdsourcing-Anbieter dargestellt werden. Eine Kampagne soll dabei in einer XML-Datei beschrieben werden können. Ein entsprechendes Format zur Auszeichnung wird im weiteren Verlauf dieses Abschnitts entwickelt.

3.3.1 Natürlichsprachliche Definition

Zunächst soll festgehalten werden, was eine Crowdsourcing-Kampagne im Sinne dieser Arbeit ausmacht. Basierend auf den in Abschnitt 3.1 vorgestellten Elementen einer Kampagne wird dazu folgende Definition aufgestellt:

> **Definition 1.** Eine Crowdsourcing-Kampagne ist eine von einem Crowdsourcer spezifizierte, möglicherweise zeitlich begrenzte, Crowdsourcing-Aktivität, bei der Vorgaben darüber gemacht werden können, Crowdsourcing-Teilnehmer in Abhängigkeit von der Erfüllung von Target-Kriterien und der Beantwortung von Ground-Truth-Fragen unterschiedlich stark zu belohnen, um dadurch Anreize für geeignete Teilnehmer zu schaffen, an dem Crowdsourcing-Unterfangen teilzunehmen und um die Qualität der Contributions aufgrund der Auswahl geeigneter Teilnehmer zu erhöhen.

Eine solche Kampagne soll unabhängig von der Crowdsourcing-Aufgabe selbst definiert werden. Stattdessen sollen Kampagnen beliebig auf bestehende Crowdsourcing-Aufgaben aufgeschaltet werden können, wenn der Crowdsourcer Bedarf dafür sieht.

3.3.2 XML-Format zur Kampagnendefinition

Crowdsourcing-Kampagnen sollen in einer einfachen XML-Datei definierbar sein. Ob der Crowdsourcer diese Datei manuell erstellt oder mithilfe einer grafischen Oberfläche generieren kann ist nicht der Fokus dieser Arbeit. Im Folgenden wird dieses XML-Format detailliert vorgestellt.

Das Wurzelelement einer Kampagnendefinition ist `campaign`, mit den optionalen Attributen `start-date` und `end-date`, um die Kampagne zeitlich einzuschränken. Innerhalb des `campaign`-Elements gibt es die drei (optionalen) Elemente `ground-truth`, `target` und `reward`. Das Grundgerüst einer XML-Kampagnendefinition sieht also wie in Listing 3.1 aus.

```
<?xml version="1.0" encoding="utf-8"?>
<cc:campaign xmlns:cc="http://rn.inf.tu-dresden.de/SANE"
        start-date="2015-07-12" end-date="2015-08-31">
    <cc:ground-truth>
        <!-- Liste von Ground-Truth-Fragen -->
    </cc:ground-truth>

    <cc:target>
        <!-- Target-Definition mit Diskriminatoren -->
    </cc:target>

    <cc:reward>
        <!-- Berechnungsvorschrift für Belohnung der Teilnehmer -->
    </cc:reward>
</cc:campaign>
```

Listing 3.1: Grundgerüst einer Crowdsourcing-Kampagnendefinition in XML

Auf die Struktur der drei inneren Elemente wird nachfolgend genauer eingegangen.

Ground Truth

Das `ground-truth`-Element trägt das optionale Attribut `score-threshold`, mit dem der vom Teilnehmer zu erreichende Ground-Truth-Mindest-Score spezifiziert werden kann. Sollte der Teilnehmer durch seine Antworten auf die Ground-Truth-Fragen diesen Score nicht erreichen, wird sein Ground-Truth-Gesamt-Score automatisch auf 0 gesetzt, so als hätte er überhaupt keine Punkte erzielt. Auf diese Weise können ungeeignete Teilnehmer leichter herausgefiltert werden.

Innerhalb des `ground-truth`-Elements befinden sich `question`-Elemente, die jeweils eine Ground-Truth-Frage repräsentieren. Die `question`-Elemente enthalten ihrerseits den Fragetext (Element `text`) sowie eine beliebige Anzahl Antwortmöglichkeiten (Element `answer`). Fragen tragen außerdem ihren Maximal-Score im Attribut `score-max`, Antworten haben ein optionales Attribut `correct` mit den möglichen Werten `true` oder `false` (standardmäßig `false`), das anzeigt, ob diese Antwortmöglichkeit korrekt ist oder nicht. Listing 3.2 zeigt beispielhaft die Definition einer Ground-Truth-Frage in diesem XML-Format.

```
<cc:question score-max="9">
    <cc:text>Welche der folgenden Tennisspieler/-innen haben mindestens
    einmal das Einzel-Tennisturnier in Wimbledon gewonnen?</cc:text>
    <cc:answer>Kevin Curren</cc:answer>
    <cc:answer correct="true">Goran Ivanišević</cc:answer>
    <cc:answer correct="true">Andy Murray</cc:answer>
    <cc:answer>Ivan Lendl</cc:answer>
    <cc:answer correct="true">Stefan Edberg</cc:answer>
    <cc:answer>Arantxa Sánchez Vicario</cc:answer>
</cc:question>
```

Listing 3.2: Beispiel-Definition einer Ground-Truth-Frage

Target

Auch für das `target`-Element gibt es das optionale Attribut `score-threshold`. Innerhalb des Targets soll der Crowdsourcer seine Anforderungen an die Teilnehmer mittels der in Abschnitt 3.2 konzipierten Diskriminatoren ausdrücken können. Es stehen die folgenden Diskriminatorelemente zur Verfügung:

- `age` für den Diskriminator *Alter*
- `time` für den Diskriminator *Zeitpunkt*
- `location` für den Diskriminator *Ort*
- `education` für den Diskriminator *Bildung/Ausbildung*
- `languages` für den Diskriminator *Sprachkenntnisse*
- `reward-points` für den Diskriminator *Belohnungspunkte*

Jedes Diskriminatorelement kann im Attribut `score-dist` die verwendete Score-Verteilung angeben. Mögliche Werte sind `binary` und `linear`, Standardwert ist `binary`. Das Attribut `score-max` gibt den Maximal-Score an, der für den Diskriminator zu erreichen sein soll, sofern dieser numerisch darstellbar ist. Ansonsten wird pro Diskriminatorwert ein eigener Score vergeben. Bei linearen Verteilungen kann außerdem im Attribut `score-min` der Minimal-Score für die Diskriminatorgrenzwerte angegeben werden. Wird das Attribut weggelassen, wird der Default-Wert 0 verwendet.

Innerhalb des jeweiligen Diskriminatorelements stehen bei den numerischen Diskriminatoren (Alter, Ort und Belohnungspunkte) die Elemente `min`, `max` und `mean` für die Diskriminatorgrenzwerte und den Zentralwert (letzterer natürlich nur bei linearer Score-Verteilung) zur Verfügung. Aufgrund der unterschiedlichen Natur der einzelnen Diskriminatoren unterscheiden sich auch die XML-Formate relativ stark, weswegen im Folgenden auf jeden einzelnen Diskriminator und dessen XML-Format eingegangen wird.

Alter. Für das Kriterium *Alter* gilt uneingeschränkt alles, was im vorhergehenden Abschnitt zu numerischen Diskriminatoren gesagt wurde. Ein Beispiel für eine Target-Definition in Bezug auf das Alter des Teilnehmers mit linearer Score-Verteilung könnte wie in Listing 3.3 aussehen.

```
<cc:age score-max="5" score-min="2" score-dist="linear">
    <cc:min>20</cc:min>
    <cc:mean>25</cc:mean>
    <cc:max>30</cc:max>
</cc:age>
```

Listing 3.3: Target-Definition für den Diskriminator *Alter* mit linearer Score-Verteilung, den Diskriminatorgrenzwerten $d_{min} = 20$ und $d_{max} = 30$, dem Zentralwert $d_{zentral} = 25$, dem Maximal-Score $s_{max} = 5$ und dem Minimal-Score $s_{min} = 2$

Ist eine binäre Score-Verteilung gewünscht, entfällt sowohl das Attribut `score-min` als auch das Element `mean`, wie Listing 3.4 zeigt.

```
<cc:age score-max="5" score-dist="binary">
    <cc:min>20</cc:min>
    <cc:max>30</cc:max>
</cc:age>
```

Listing 3.4: Target-Definition für den Diskriminator *Alter* mit binärer Score-Verteilung, den Diskriminatorgrenzwerten $d_{min} = 20$ und $d_{max} = 30$ und dem Maximal-Score $s_{max} = 5$

Zeit. Auch der Zeitpunkt der Contribution soll sowohl binäre als auch lineare Verteilungen erlauben. Zur größeren Flexibilität soll der Crowdsourcer dieses Target-Kriterium auf zwei unterschiedliche Weisen angeben können: als einmalige Zeitspanne mit genauer Datums- und Zeitangabe des Beginn- und Endzeitpunktes oder als Zeitspanne zwischen zwei Uhrzeiten, die wiederkehrend für jeden Tag gilt. In letzterem Fall sollen außerdem einzelne Wochentage ausgeschlossen werden können, auf die der Diskriminator nicht abzielen soll.

Listing 3.5 zeigt ein beispielhaftes Targeting für Contributions innerhalb eines Zeitraums an einem konkreten Datum. Generell sollen Datums- und Uhrzeitformate immer den Empfehlungen von ISO 8601 folgen. Jahr, Monat und Tag des Monats sind durch je ein Minuszeichen getrennt und zwischen Datum und Uhrzeit steht ein großes „T“. Am Ende der Zeitangabe folgt die Angabe über die Zeitzone, das „Z“ steht dabei für UTC[4].

```
<cc:time score-max="10" score-min="1" score-dist="linear">
    <cc:min>2015-01-15T13:00:00Z</cc:min>
    <cc:mean>2015-01-15T17:00:00Z</cc:mean>
    <cc:max>2015-01-15T21:00:00Z</cc:max>
</cc:time>
```

Listing 3.5: Target-Definition für Contributions zwischen dem 15.01.2015, 13 Uhr und dem 15.01.2015, 21 Uhr mit linearer Score-Verteilung

Soll nicht nur auf Contributions an einem bestimmten Tag abgezielt werden, sondern auf Beiträge, die werktags zwischen 13 und 21 Uhr abgegeben werden, könnte die Target-Definition dafür wie in Listing 3.6 aussehen.

```
<cc:time score-max="10" score-min="1" score-dist="linear">
    <cc:min>13:00:00Z</cc:min>
    <cc:mean>17:00:00Z</cc:mean>
    <cc:max>21:00:00Z</cc:max>
    <cc:exclude-weekdays>
        <cc:weekday>Saturday</cc:weekday>
        <cc:weekday>Sunday</cc:weekday>
    </cc:exclude-weekdays>
</cc:time>
```

Listing 3.6: Target-Definition für Contributions, die von Montag bis Freitag jeweils zwischen 13 und 21 Uhr abgegeben werden, mit linearer Score-Verteilung

Die Datumsangabe kann also weggelassen werden, womit das Kriterium dann automatisch jeden Tag greifen würde. Durch das explizite Ausschließen des Wochenendes fallen allerdings nur Contributions an Werktagen in das Target-Schema. Bei einer gewünschten binären Verteilung würden das `score-min`-Attribut und das `mean`-Element wegfallen, der Rest bliebe unverändert.

Ort. Der Diskriminator *Ort* ist ein gewisser Sonderfall, da hier der eigentliche Diskriminatorwert, der mit einem Target-Score versehen werden soll, die räumliche Distanz zwischen der Teilnehmer-Location und der in der Kampagnendefinition angegebenen Target-Location ist. Aus diesem Grund enthält, unabhängig von der verwendeten Score-Verteilung, das `location`-Element der Kampagnendefinition ein Element `mean` mit einer genauen Target-Location. Außerdem gibt es das Element `max-distance` mit der maximalen Entfernung von dieser Target-Location (in km), die (wie in Abschnitt 3.2 dargelegt) mindestens das Doppelte des verwendeten Verschiebungsradius m betragen muss. Listing 3.7 zeigt ein Beispiel für eine derartige Target-Definition.

[4]Coordinated Universal Time (Koordinierte Weltzeit)

```
<cc:location score-max="10" score-min="5" score-dist="linear">
    <cc:mean>
        <cc:lat>43.014252</cc:lat>
        <cc:lon>-3.209216</cc:lon>
    </cc:mean>
    <cc:max-distance>200</cc:max-distance>
</cc:location>
```

Listing 3.7: Target-Definition für den Diskriminator *Ort* mit linearer Score-Verteilung in einem Umkreis von 200 km um eine Target-Location

Für eine binäre Verteilung müsste nur das Attribut `score-min` weggelassen und der Wert von `score-dist` entsprechend geändert werden. Dann würde für alle möglichen Orte im Umkreis von 200 km um die in `mean` angegebene Location der Score `score-max` ausgeschüttet werden, statt ihn linear zu interpolieren.

Bildung/Ausbildung. Für diesen Diskriminator gibt es das Element `education`. Darin ist mit je einem `qualification`-Element jeder in der Kampagne angestrebte, höchste Bildungsabschluss des Teilnehmers zusammen mit dem dazugehörigen Score gelistet. Listing 3.8 zeigt eine solche beispielhafte Target-Definition und alle zur Verfügung stehenden Diskriminatorwerte in den `name`-Attributen der `qualification`-Elemente.

```
<cc:education score-dist="binary">
    <cc:qualification name="none" score="0" />
    <cc:qualification name="hauptschule" score="1" />
    <cc:qualification name="mittlere_reife" score="1" />
    <cc:qualification name="abitur" score="2" />
    <cc:qualification name="ausbildung" score="2" />
    <cc:qualification name="bachelor" score="5" />
    <cc:qualification name="master" score="10" />
    <cc:qualification name="promotion" score="15" />
    <cc:qualification name="habilitation" score="15" />
</cc:education>
```

Listing 3.8: Target-Definition für den Diskriminator *Bildung/Ausbildung* mit unterschiedlichen Scores pro Diskriminatorwert

Da nur binäre Score-Verteilungen möglich sind, kann das Attribut `score-dist` auch weggelassen werden, aber auf keinen Fall den Wert `linear` enthalten.

Sprachkenntnisse. Die Sprachkenntnisse, auf die die Kampagne abzielt, sollen ebenso wie die Bildungsabschlüsse in der XML-Datei aufgezählt und einzeln mit einem Score versehen werden. Die Sprachen werden dabei durch ihren zweibuchstabigen Code nach ISO 639-1 [Int67] angegeben. Soll eine (oder mehrere) der Sprachen die Muttersprache des Teilnehmers sein, so kann dies mit dem Attribut `native` angezeigt werden. So könnte eine Kampagnendefinition, die auf englische Muttersprachler, die außerdem Spanisch und Portugiesisch sprechen, wie in Listing 3.9 dargestellt aussehen.

```
<cc:languages score-dist="binary">
    <cc:language code="en" native="true" score="10" />
    <cc:language code="pt" score="6" />
    <cc:language code="es" score="5" />
</cc:languages>
```

Listing 3.9: Target-Definition für den Diskriminator *Sprachkenntnisse*, die auf englische Muttersprachler mit Kenntnissen des Spanischen und Portugiesischen abzielt

Belohnungspunkte. Bei den Belohnungspunkten handelt es sich wieder um einen Spezialfall, da man hier in der Regel den Diskriminatorwert nach oben nicht begrenzen möchte bzw. kann. Die Kampagnendefinition soll es zwar erlauben, auch eine Obergrenze festzulegen, falls dies für bestimmte Anwendungen sinnvoll ist. Im Allgemeinen jedoch wird der Crowdsourcer nach der Regel „je mehr, desto besser“ verfahren und für die Anzahl der Belohnungspunkte höchstens eine Untergrenze angeben wollen.

Damit trotzdem lineare Score-Verteilungen möglich sind, muss zusätzlich zum Minimalwert für die Anzahl der Belohnungspunkte ein weiterer Diskriminator-Referenzwert d_{ref} angegeben werden, für den ein Referenz-Score s_{ref} generiert wird. Aus den so entstehenden beiden Punkten (d_{min}, s_{min}) und (d_{ref}, s_{ref}) entlang der linearen Verteilung kann dann mittels Inter- bzw. Extrapolation der Score für jeden beliebigen Diskriminatorwert berechnet werden. Aus diesem Grund muss – wenn die Anzahl der Belohnungspunkte im Target nach oben unbegrenzt sein soll – im Attribut `score-reference` des Elements `reward-points` der Referenz-Score angegeben werden und innerhalb von `reward-points` ein Element `reference` existieren, dass den Referenz-Diskriminatorwert angibt, der diesen Referenz-Score erzeugen soll. Listing 3.10 verdeutlicht diesen Fall an einem Beispiel.

```
<cc:reward-points score-reference="40" score-min="10" score-dist="linear">
    <cc:min>20</cc:min>
    <cc:reference>100</cc:reference>
</cc:reward-points>
```

Listing 3.10: Target-Definition für eine geforderte Mindestanzahl von 20 Belohnungspunkten, wobei für diese Mindestanzahl der Mindestscore von 10 und für einen weiteren Referenzwert von 100 Belohnungspunkten ein Score von 40 vergeben werden soll; Darüber hinausgehende Werte werden extrapoliert.

Im wahrscheinlich seltenen Fall, dass die Diskriminatorwerte für die Belohnungspunkte doch in beide Richtungen begrenzt sein sollen, kann das Element `reward-points` im XML wie gehabt mit den Attributen `score-min` und `score-max` ausgestattet werden und die Unterelemente `min`, `mean` und `max` haben.

Bei binären Verteilungen, bei denen der Diskriminatorwert – also die Anzahl der Belohnungspunkte – nach oben unbegrenzt sein soll, vereinfacht sich die Notation in der Kampagnendefinition, da die Interpolation über einen Referenzwert hier im Gegensatz zu einer linearen Verteilung entfällt. Es muss dann lediglich das Element `min` innerhalb des `reward-points`-Elements sowie das Attribut `score-max` angegeben werden, wie Listing 3.11 zeigt.

```
<cc:reward-points score-max="40" score-dist="binary">
    <cc:min>20</cc:min>
    <!-- kein Element 'max', da nach oben unbegrenzt -->
</cc:reward-points>
```

Listing 3.11: Target-Definition für eine geforderte Mindestanzahl von 20 Belohnungspunkten, ab der wegen der binären Score-Verteilung der Maximal-Score von 40 vergeben werden soll

Natürlich kann auch hier das `max`-Element angegeben werden, wenn die im Target angestrebte Anzahl an Belohnungspunkten aus irgendwelchen Gründen nach oben begrenzt sein soll.

Reward

Im `reward`-Element der Kampagnendefinition gibt es das Element `formula`, dass die in Abschnitt 3.1.3 erläuterte Berechnungsformel zur Belohnung des Teilnehmers für seine Contribution enthält. Innerhalb dieser Formel stehen die Variablen `$targetScore` und `$groundTruthScore` – soweit die entsprechenden Elemente in der Kampagnendefinition existieren – und die vier grundlegenden mathematischen Operatoren `+`, `-`, `*` und `/` zur Verfügung. Außerdem können Zahlen mit Punkt als Dezimaltrennzeichen verwendet sowie Ausdrücke geklammert werden. Eine solche Berechnungsformel innerhalb des Reward-Abschnitts kann z. B. wie in Listing 3.12 gezeigt aussehen.

```
<cc:reward>
    <cc:formula>($targetScore + $groundTruthScore) * 1.5</cc:formula>
</cc:reward>
```

Listing 3.12: Berechnungsvorschrift zur Belohnung eines Teilnehmers im Reward-Teil der Kampagnendefinition

3.4 Zusammenfassung

Eine Crowdsourcing-Kampagne wurde definiert als Crowdsourcing-Aktivität, bei der der Crowdsourcer Vorgaben darüber macht, nach welchem Schema Teilnehmer als geeignet eingestuft und wie sie belohnt werden sollen. Das Konzept sieht für eine Kampagne drei wesentliche Komponenten vor. Zunächst gibt es das Target, in dem mithilfe von Diskriminatoren ein Filter für geeignete Teilnehmer angegeben wird. Optional enthält die Kampagne weiterhin eine Liste mit Ground-Truth-Fragen zur Ermittlung des Fachwissens der Teilnehmer. Schließlich gibt es den Reward-Abschnitt der Kampagnendefinition, in dem bestimmt wird, wie sich die Belohnungspunkte eines Teilnehmers in Abhängigkeit von seiner Eignung für die Kampagne berechnen.

Um diese Eignung jeweils zu objektivieren, wird sowohl im Target- als auch im Ground-Truth-Teil ein Score berechnet. Der Target-Score hängt davon ab, wie stark ein Teilnehmer bzw. seine Contribution den Anforderungen der Kampagne gerecht werden. Generell können für einen bestimmten Diskriminator binäre und lineare Score-Verteilungen definiert werden;

die Score-Punkte können also entweder nach dem Alles-oder-nichts-Prinzip verteilt oder linear interpoliert werden, sodass der Target-Score genauer mit der Eignung des Teilnehmers korreliert. Der Ground-Truth-Score berechnet sich aus der Qualität der Antworten eines Teilnehmers auf die Ground-Truth-Fragen, die als *Multiple Mark Questions* (MMQs) formuliert sind. Zum Scoring von MMQs empfiehlt sich – trotz einiger Ungenauigkeiten in gewissen Spezialfällen – die Formel von Tarasowa und Auer [TA13]. Target-Score und Ground-Truth-Score zusammen können dann im Reward-Teil der Kampagnendefinition benutzt werden, um in einer Berechnungsformel die Anzahl der Belohnungspunkte auszurechnen, die dem Teilnehmer für seine Contribution zustehen.

Für eine solche Kampagnendefinition wurde ein XML-Format konzipiert. Die verwendbaren Diskriminatoren sind *Alter* (des Teilnehmers), *Zeitpunkt* (der Contribution), *Ort* (des Teilnehmers zum Zeitpunkt der Contribution), *Bildung/Ausbildung* und als weiterer Spezialfall *Sprachkenntnisse* sowie die bisherige Anzahl *Belohnungspunkte* des Teilnehmers. Weiterhin erwähnenswert ist, dass der Ort des Teilnehmers aus Datenschutzgründen nur obfuskiert gespeichert wird. Diese Kriterien sind möglicherweise erweiterbar. Außerdem muss ihre Tauglichkeit für das Targeting geeigneter Crowdsourcing-Teilnehmer erst untersucht werden. Für diese Arbeit stellen sie aber einen guten Kompromiss aus Sinnhaftigkeit und Einfachheit der Erfassung dar.

4 Implementierung

Gemäß dem in Kapitel 3 entwickelten Konzept sollen in diesem Teil der Arbeit die Schritte zur Implementierung dieses Konzepts dargelegt werden.

4.1 Implementierung des XML-Formats zur Kampagnendefinition

Es gibt verschiedene Möglichkeiten, eine Validierung von XML-Inhalten durchzuführen. Eine eigens entwickelte Softwarekomponente dafür würde die totale Freiheit und Mächtigkeit bieten, ist aber auch mit einem enormen Implementierungsaufwand verbunden. Einfacher ist die Verwendung von XML-Schemasprachen, in denen sich die geforderte Dokumentstruktur festlegen lässt. Die bekanntesten Vertreter sind DTD und XSD.

4.1.1 DTD vs. XSD

DTD (Document Type Definition) benutzt eine Syntax zur Beschreibung der Struktur eines XML-Dokuments, die selbst zwar an XML erinnert, aber kein XML ist. XSD (XML Schema Definition) hingegen ist selbst ein XML-Dialekt, was aufgrund der weiten Verbreitung von XML wahrscheinlich in vielen Fällen zu einer leichteren Verständlichkeit eines XSD-Dokuments für einen breiten Leserkreis führen wird. Ein weiterer Vorteil besteht in der Möglichkeit, in XSD Datentypen für Elemente und Attribute vorzugeben.

Was beide Schemasprachen vermissen lassen, ist eine konditionale Definition von Elementen und Attributen sowie deren Werten. Beispielsweise ist es in der Definition einer Crowdsourcing-Kampagne wichtig, dass ein `mean`-Element nur in Diskriminatorelementen vorkommt, die für das Attribut `score-dist` den Wert `linear` angegeben haben. Diese Einschränkung lässt sich aber weder in DTD noch in XSD zufriedenstellend ausdrücken. Gleiches gilt auch für Abhängigkeiten zwischen Elementwerten von der Form „der Wert von `max` muss größer sein als der Wert von `min`“. Es bleibt für solche Fälle nur die spätere Auswertung im Programmcode, der die Kampagnendefinition interpretiert. In der Schemadefinition selbst lassen sich bestenfalls Kommentare für Entwickler hinterlegen, die auf solche Zusatzbedingungen hinweisen.

Trotz der erwähnten Einschränkungen hat XSD klare Vorteile gegenüber DTD und wird daher in dieser Arbeit zur Definition des XML-Formats für Crowdsourcing-Kampagnen verwendet. Die Einzelheiten der so entstandenen XSD-Datei werden im Folgenden vorgestellt.

4.1.2 XSD-Datei für das XML-Kampagnenformat

Die Implementierung der Schemadefinition leitet sich mehr oder weniger direkt aus dem in Abschnitt 3.3.2 vorgestellten XML-Format ab.

Grundstruktur

Unterhalb des Wurzelelements `campaign` sollen alle drei Elemente `ground-truth`, `target` und `reward` als optional deklariert werden. Allerdings soll die Einschränkung gelten, dass mindestens eines dieser Elemente vorhanden sein muss (da sonst die Kampagnendefinition wenig Sinn ergibt). Da XSD – wie bereits erwähnt – bei derlei Zusatzbedingungen oft die Mächtigkeit in der Syntax vermissen lässt, muss diese Einschränkung mit einem `xs:choice`-Konstrukt ausgedrückt werden. Die Struktur des `campaign`-Elements wurde auf diese Weise implementiert und ist in Listing 4.1 dargestellt. Man kann sich leicht vorstellen, wie schlecht ein solches Konstrukt bei mehr als drei Elementen und den daraus resultierenden Kombinationen skalieren würde.

```
<xs:element name="campaign">
    <xs:complexType>
        <xs:choice>
            <xs:sequence>
                <xs:element name="ground-truth"
                            type="tns:groundTruthType" />
                <xs:element name="target" type="tns:targetType"
                            minOccurs="0" />
                <xs:element name="reward" type="tns:rewardType"
                            minOccurs="0" />
            </xs:sequence>
            <xs:sequence>
                <xs:element name="target" type="tns:targetType" />
                <xs:element name="reward" type="tns:rewardType"
                            minOccurs="0" />
            </xs:sequence>
            <xs:element name="reward" type="tns:rewardType" />
        </xs:choice>
        <xs:attribute name="start-date" type="xs:date" />
        <xs:attribute name="end-date" type="xs:date" />
        <xs:attribute name="domain">
            <xs:simpleType>
                <xs:list itemType="tns:domainType" />
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
</xs:element>
```

Listing 4.1: Schemadefinition des Wurzelelements `campaign` einer Kampagnendefinition, mit drei Unterelementen, von denen mindestens eines auftreten muss

Das xs:choice-Element lässt genau eine der aufgelisteten Alternativen zu. Dementsprechend gibt es folgende Möglichkeiten für das Vorkommen der Kampagnenelemente, die alle möglichen Kombinationen abdecken:

- Das Element ground-truth kommt einmal vor, optional gefolgt vom Element target (wg. minOccurs="0"), optional gefolgt vom Element reward.
- Das Element target kommt einmal vor, optional gefolgt vom Element reward, aber kein Vorkommen von ground-truth.
- Nur das Element reward kommt vor.

Es ist aber mit dieser Definition unmöglich, die Reihenfolge der Elemente *nicht* vorzuschreiben, da innerhalb von xs:choice kein xs:all-, sondern nur das alternative xs:sequence-Element verwendet werden kann.

Außerdem sind für das Element campaign die Attribute start-date und end-date, jeweils mit dem Typ xs:date, definiert. Dieser Datentyp implementiert den Standard ISO 8601, der das Datumsformat vorschreibt, das laut Konzeption in den Kampagnendefinitionen verwendet werden soll. Mit dem dritten (optionalen) Attribut domain soll die Anwendungsdomäne der Kampagne angegeben werden können. Näheres dazu wird im Abschnitt „Domänenbeschränkung von Kampagnendefinitionen“ erklärt.

Besonderheiten der Kampagnenelemente

Die Typdefinitionen für ground-truth, target und reward, auf die jeweils verwiesen wird, sind an anderer Stelle innerhalb der XSD-Datei aufgeführt, um sie wiederverwendbar zu gestalten. Beim targetType stellt sich im Prinzip dasselbe Problem, das schon beim übergeordneten Element campaign auftrat. Es sollen beliebig viele, aber mindestens eines der Unterelemente vorkommen (und idealerweise in beliebiger Reihenfolge). Bei den sechs konzipierten Diskriminatorelementen wird der Aufwand für das dazu nötige xs:choice-Konstrukt allerdings unvertretbar. Dazu kommt wiederum, dass unterhalb von xs:choice ohnehin kein xs:all, sondern nur xs:sequence auftreten darf, also die Reihenfolge der Elemente genau vorgeschrieben werden muss. Aus demselben Grund dürfen auch ground-truth, target und reward nur in dieser Reihenfolge vorkommen.

Aufgrund dessen wurden alle Definitionen der Diskriminatorelemente in einem xs:all-Block zusammengefasst und mit dem Attribut minOccurs="0" (i. e. als optional) ausgezeichnet. Damit beschränken sich die Nachteile, die durch die Limitierungen von XSD entstehen, auf ein Minimum: Es ist jetzt möglich, ein leeres Target-Element ohne Diskriminatoren anzugeben. Auf der anderen Seite können aufgrund des xs:all die Diskriminatoren in beliebiger Reihenfolge aufgelistet werden.

Erwähnenswert ist das Diskriminatorelement time. Damit hier – wie konzipiert – sowohl Datums- als auch Zeitangaben gemacht werden können, müssen die entsprechenden Datentypen xs:time und xs:dateTime vereinigt werden. Dazu existiert in XSD das Element xs:union, mit dem ein neuer Datentyp aus einer Liste von existierenden Datentypen erzeugt wird. Die so entstandene Typdefinition lässt dann alle Werte zu, die einem der beiden beteiligten Typen entsprechen. Der XSD-Code für diesen vereinigten Datentyp timeType zeigt Listing 4.2.

```
<xs:simpleType name="timeType">
    <xs:union memberTypes="xs:time xs:dateTime" />
</xs:simpleType>
```

Listing 4.2: Typdefinition für vereinigten Datentyp, der Datums- und Zeitangaben ermöglichen soll

Auf diesen so definierten Datentyp `timeType` wird dann vom `min`-, `max`- und `mean`-Element des Diskriminatorelements `time` aus verwiesen.

Generell wurde stets versucht, alle in XSD mit vertretbarem Aufwand ausdrückbaren Einschränkungen und Abhängigkeiten zwischen den Elementen und ihren Werten festzulegen. Oftmals ist es aber schlicht unmöglich, alle Bedingungen dieser Art abzubilden. Dazu gehört die bereits erwähnte Abhängigkeit zwischen dem Vorkommen des `mean`-Elements und dem Attributwert `score-dist="linear"`, die in XSD nicht ausgedrückt werden kann. Ebenfalls zumindest kompliziert ist das Vorschreiben des Formats für die Berechnungsformel der Belohnungspunkte im `reward`-Element. Auf die Benutzung eines regulären Ausdrucks zu diesem Zweck wurde zugunsten einer einfacheren Definition als `xs:string` verzichtet. Die Schemadefinition des `reward`-Elements der Kampagnendefinition, die sich damit ergibt, zeigt Listing 4.3.

```
<xs:complexType name="rewardType">
    <xs:all>
        <xs:element name="formula" type="xs:string" />
    </xs:all>
</xs:complexType>
```

Listing 4.3: Schemadefinition des `reward`-Elements der Kampagnendefinition

Domänenbeschränkung von Kampagnendefinitionen

Eine Crowdsourcing-Kampagne soll immer zusätzlich zu einem bestehenden Crowdsourcing definiert werden. Ein Crowdsourcing (im Sinne von SANE) bietet hierbei verschiedene Methoden an, die von den Teilnehmern (bzw. deren Crowdsourcing-Clients) aufgerufen werden, um Contributions abzusenden. Eine Crowdsourcing-Kampagne soll nun zusätzlich mit einer Liste von *Domänen* versehen werden können, die ihre Geltungsbereiche angibt. Eine Domäne folgt dabei immer dem Muster `/<NameDesCrowdsourcings>/<NameDerMethode>`. Für beide Platzhalter können mit einem Asterisk Wildcards vergeben werden, die auf alle Crowdsourcings bzw. auf alle Methodennamen passen. Beispiele für gültige Domänenangaben und ihre Semantik sind im Folgenden aufgelistet:

`/MapBiquitous/createWLANFingerprint`
: Die Kampagne gilt für die Methode `createWLANFingerprint` des Crowdsourcings `MapBiquitous`.

`/MapBiquitous/*`
: Die Kampagne gilt für alle Methoden des Crowdsourcings `MapBiquitous`.

`/*/createWLANFingerprint`
: Die Kampagne gilt für die Methode `createWLANFingerprint` jedes Crowdsourcings.

`/*/*`
: Die Kampagne gilt für alle Methoden jedes Crowdsourcings.

Dabei sollte in irgendeiner Form sichergestellt werden, dass ein Crowdsourcer eine Kampagne nicht für Crowdsourcings definiert, bei denen er dazu keine Berechtigung hat. Wie solch ein Mechanismus funktionieren könnte, wird innerhalb dieser Arbeit nicht weiter behandelt und zukünftigen Erweiterungen überlassen.

Umgesetzt werden soll die Domänenbeschränkung mit dem bereits in Listing 4.1 aufgeführten Attribut `domain` im `campaign`-Element. Es soll eine Liste von Werten enthalten, deren Datentyp jeweils auf `xs:string` basiert und diesem eine Beschränkung mithilfe eines regulären Ausdrucks hinzufügt, die nur Angaben im oben erwähnten Format zulässt. Listing 4.4 zeigt die entsprechende Typdefinition von `domainType`.

```
<xs:simpleType name="domainType">
    <xs:restriction base="xs:string">
        <xs:pattern value="(/(\*|[A-Za-z][A-Za-z0-9_-]*)){2}" />
    </xs:restriction>
</xs:simpleType>
```

Listing 4.4: Definition des XML-Datentyps für eine Domänenangabe einer Crowdsourcing-Kampagne

Zusammenfassung

Es ist eine XSD-Datei entstanden, die Kampagnendefinitionen im konzipierten XML-Format validieren kann. Einige Einschränkungen konnten aufgrund der Syntax von XSD allerdings nicht vollumfänglich abgebildet werden. Für solche Fälle bleibt jedoch die nachträgliche Prüfung im Programmcode, sodass insgesamt trotzdem alle in Kapitel 3 genannten Elemente einer Crowdsourcing-Kampagne implementiert werden können. Zusätzlich zu den konzipierten Elementen kam die Angabe einer Domäne (i. e. eines Gültigkeitsbereichs) einer Kampagne hinzu. Darin kann festgelegt werden, für welche Methode(n) aus welchem Crowdsourcing die Kampagne gelten soll.

4.1.3 Kampagnenvalidierung im Programmcode

Um eine von einem Crowdsourcer bereitgestellte Kampagnendefinition möglichst vollständig zu validieren, reicht die Schemadefinition in XSD nicht aus. XSD ist nicht mächtig genug, um Abhängigkeiten zwischen Werten und Vorkommen von Attributen und Elementen auszudrücken. Diese müssen, falls sie ebenfalls validiert werden sollen, daher im Programmcode implementiert werden.

In XSD nicht formulierbare Constraints

Im konzipierten XML-Format zur Definition von Crowdsourcing-Kampagnen wurden 14 solcher Einschränkungen identifiziert, die im Folgenden aufgelistet sind:

1. Im `campaign`-Element darf das Datum im Attribut `start-date` zeitlich nicht nach dem im Attribut `end-date` liegen.
2. Im `ground-truth`-Element muss jedes `question`-Element mindestens ein `answer`-Element haben, das im Attribut `correct` den Wert `true` hat.
3. Im `target`-Element muss mindestens ein Diskriminatorelement vorkommen.
4. Beim Diskriminatorelement `time` müssen die Datums- bzw. Zeitangaben in den Elementen `min`, `max` und (falls vorhanden) `mean` alle im selben Format vorliegen, also entweder als kombinierte Datums- und Zeitangabe oder als reine Zeitangabe.
5. Bei den Elementen `age`, `time` und `reward-points` müssen die Ungleichungen $d_{min} < d_{max}$ und $d_{min} \leq d_{zentral} \leq d_{max}$ und beim Element `reward-points` zusätzlich $d_{min} < d_{reference}$ erfüllt sein (also für die Werte, die in den Elementen `min`, `mean`, `max` und `reference` stehen).
6. Bei den Elementen `age`, `time` und `reward-points` muss gelten:
   a) Wenn das Attribut `score-dist` den Wert `linear` hat, dann muss auch das Attribut `score-min` und bei den Elementen `age` und `time` zusätzlich auch das Element `mean` vorkommen.
   b) Wenn das Attribut `score-dist` nicht vorkommt oder nicht den Wert `linear` hat, dann dürfen weder das Attribut `score-min` noch das Element `mean` vorkommen.
7. Im Element `time` darf innerhalb von `exclude-weekdays` ein `weekday`-Element mit demselben Wochentag als Wert nicht mehrfach vorkommen.
8. Im Element `location` muss der Wert des Elements `max-distance` mindestens das Doppelte des Obfuskationsradius m betragen.
9. Bei den Elementen `age`, `time`, `location` und `reward-points` muss der Wert des Attributs `score-min` kleiner als der Wert des Attributs `score-max` sein, sofern beide vorhanden sind.
10. Im Element `education` dürfen nicht mehrere `qualification`-Elemente mit demselben Wert im `name`-Attribut vorkommen.
11. Im Element `languages` dürfen nicht mehrere `language`-Elemente mit derselben Kombination aus Werten für die Attribute `code` und `native` vorkommen. Dabei muss auch beachtet werden, dass ein weggelassenes `native`-Attribut und `native="false"` semantisch identisch sind.
12. Beim Element `reward-points` muss gelten: Wenn das Attribut `score-dist` nicht vorkommt oder nicht den Wert `linear` hat, dann dürfen weder das Element `reference` noch das Attribut `score-reference` auftreten.
13. Beim Element `reward-points` muss gelten: Wenn das Attribut `score-dist` den Wert `linear` hat, dann müssen *entweder* die Elemente `mean` und `max` und das Attribut

`score-max` *oder* das Element `reference` und das Attribut `score-reference` vorkommen.

14. In der Berechnungsformel im Element `formula` innerhalb des Elements `reward` dürfen keine einfachen oder doppelten Anführungsstriche und keine Bezeichner außer `$targetScore` und `$groundTruthScore` (und diese auch nur bei Vorhandensein des entsprechenden Elements in der Kampagnendefinition) vorkommen und sie muss darüber hinaus ein numerisches Resultat liefern.

Implementierung der Kampagnenvalidierung in PHP

Um diese Zusatzbedingungen zu validieren, wurde eine PHP-Klasse namens `CampaignValidator` angelegt und darin die Validierung implementiert. Für jeden der Constraints gibt es eine Methode in dieser Klasse, die bei erfolgreicher Validierung `true`, ansonsten eine natürlichsprachliche Fehlermeldung des ersten aufgetretenen Problems zurückgibt. In Gang gesetzt werden kann die Validierung durch den Aufruf der Methode `checkValidity` in der Klasse `CampaignValidator`, die ihrerseits intern alle Methoden für die einzelnen Constraints aufruft und am Ende ein Objekt der Klasse `ValidationResult` zurückliefert. Dieses zeigt das Ergebnis des Validierungsprozesses an und stellt es über die beiden Methoden `isValid` und `getErrorList` nach außen zur Verfügung. Letztere liefert ggf. eine Liste aller Fehlermeldungen, die die Methoden zur Validierung der einzelnen Constraints produziert haben. Um auf den Elementbaum in der XML-Datei mit der Kampagnendefinition zuzugreifen, wurde die in PHP integrierte Programmierbibliothek *SimpleXML*[1] verwendet.

An den Stellen, an denen Datums- und Zeitangaben in der Kampagnendefinition ausgewertet werden müssen, geschieht das über die Standard-PHP-Klasse `DateTime`, die sowohl eine Konstante `ISO8601` mit einem entsprechenden Format-String als auch eine Methode `createFromFormat` anbietet. Trotzdem reicht es nicht aus, nur diese Bordmittel von PHP zu verwenden, um alle zu unterstützenden Formate zu validieren, da der Format-String in der `ISO8601`-Konstante zwingend die Angabe eines Datums vorschreibt. Listing 4.5 zeigt die zusätzlich implementierte Hilfsmethode `getDateTimeFromISOString`, die diese Limitierung umgeht, indem sie einen zusätzlichen Format-String für reine Zeitangaben implementiert.

```
public static function getDateTimeFromISOString($isoString) {
    $result = DateTime::createFromFormat(DateTime::ISO8601, $isoString);
    if (!$result) {
        $result = DateTime::createFromFormat('H:i:sT', $isoString);
    }

    return $result;
}
```

Listing 4.5: Methode zum Erzeugen eines `DateTime`-Objekts aus ISO-8601-kompatiblen Datums- und Zeitangaben

[1] `http://php.net/manual/en/book.simplexml.php`

Auf diese Weise können auch Zeitangaben von der Form `13:37:42+01:00` oder `13:37:42Z` in gültige `DateTime`-Objekte umgewandelt werden. Die Formatoption `T` sorgt für die korrekte Interpretation der Zeitzone.

Ein weiterer erwähnenswerter Aspekt der Kampagnenvalidierung ist das Prüfen der Berechnungsformel für die Belohnungspunkte. Diese Prüfung erfolgt im Wesentlichen in drei Schritten:

1. Es muss sichergestellt werden, dass die Formel keine einfachen oder doppelten Anführungsstriche enthält, die auf Strings hindeuten.
2. Es wird in der Formel nach Bezeichnern gesucht. Die einzigen gültigen Bezeichner, die vorkommen dürfen, sind `$targetScore` und `$groundTruthScore`, allerdings auch nur, wenn es in der Kampagnendefinition ein Element `target` bzw. `ground-truth` gibt. Die Suche nach Bezeichnern erfolgt mithilfe eines regulären Ausdrucks, der von offizieller Seite so angegeben wird [The15]. Listing 4.6 zeigt den entsprechenden Aufruf an die Methode `preg_match_all`, die eine Zeichenkette nach auf einen regulären Ausdruck passenden Vorkommen untersucht und alle Treffer in einem Array ablegt.

```
preg_match_all("/[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*/",
    $formula, $hits);
```

Listing 4.6: Suche nach in PHP gültigen Bezeichnern innerhalb der Berechnungsformel für die Belohnungspunkte

3. Sind diese beiden Vorbedingungen sichergestellt, kann das testweise Auswerten der Formel mithilfe der PHP-Funktion `eval` passieren. Diese führt beliebigen PHP-Code, der ihr als Zeichenkette übergeben wird, aus und ist daher – gerade, wenn der Code aus Benutzereingaben stammt – mit äußerster Vorsicht zu genießen. Listing 4.7 zeigt, wie das Auswerten der Formel im Code aussieht.

```
$targetScore = 1;
$groundTruthScore = 1;

ob_start();
$evalResult = eval("\$formulaResult = $formula;");

if ('' !== ob_get_clean()) {
    // Error while evaluating
    return 'The reward formula is not evaluable.';
}

if (is_numeric($formulaResult)) {
    return true;
} else {
    return 'The result of the reward formula is not numeric.';
}
```

Listing 4.7: Testweises Auswerten der Berechnungsformel für die Belohnungspunkte mit der PHP-Funktion `eval`

Damit die Auswertung durch den `eval`-Aufruf ein numerisches Resultat liefern kann, werden also zunächst die Variablen `$targetScore` und `$groundTruthScore`, die innerhalb der Formel vorkommen dürfen, mit einem numerischen Wert (in diesem Fall der Zahl 1) vorbelegt. Da mit den Schritten 1 und 2 bereits sichergestellt wurde, dass in der Formel keine Strings und keine Identifier enthalten sind, sollte es an dieser Stelle einigermaßen sicher sein, den `eval`-Aufruf zu machen. Da die Formel aber nach wie vor Syntaxfehler (z. B. ungültige Aneinanderreihungen von Rechenzeichen, Zahlen und/oder Klammern) enthalten kann und PHP diese genauso wie einen Syntaxfehler im eigentlichen Programmcode behandeln und entsprechend mit Fehlermeldungen signalisieren würde, muss vorher mittels der `ob_start`-Methode die Ausgabepufferung aktiviert werden. Dadurch werden eventuelle Parser-Fehlermeldungen in einen Ausgabepuffer geschrieben, statt sie direkt an den Client weiterzugeben.

Danach erfolgt der Aufruf von `eval`. Als auszuführender Code wird eine Zuweisung an die Variable `$formulaResult` übergeben. Diese enthält nach der erfolgreichen Ausführung den Wert, zu dem der Ausdruck in `$formula` evaluiert wurde. Anschließend folgt noch die Prüfung, ob etwas in den Ausgabepuffer geschrieben wurde, i. e. ob der `eval`-Aufruf Fehler produziert hat. `ob_get_clean` liefert den Inhalt des Ausgabepuffers als String und leert ihn gleichzeitig. War der Ausgabepuffer leer, muss nur noch überprüft werden, ob das Ergebnis der Auswertung der Berechnungsformel numerisch ist. Es wären beispielsweise auch boolesche Werte möglich, denn die Formel kann auch logische Operatoren und Vergleichsoperatoren enthalten und trotzdem alle bisherigen Prüfungen überstanden haben.

Sollte die Validierung der Berechnungsformel bereits bei Schritt 1 oder 2 scheitern, wird in der Methode sofort eine Beschreibung des Problems als String zurückgegeben und es kommt gar nicht erst zur Prüfung der Auswertbarkeit in Schritt 3. Somit ist garantiert, dass der `eval`-Aufruf nur dann gemacht wird, wenn Strings und Bezeichner – also die potentiellen Sicherheitsrisiken – innerhalb der Formel nicht vorkommen.

4.2 Implementierung der Kampagnenlogik

Ebenfalls in PHP implementiert wurde die benötigte Geschäftslogik einer Crowdsourcing-Kampagne. Dazu gehört die Berechnung von Grount-Truth- und Target-Scores sowie der Anzahl der Belohnungspunkte. Letzteres wurde wiederum mithilfe der PHP-Funktion `eval` realisiert. Außerdem wurde die Obfuskation der Teilnehmer-Location umgesetzt.

Insgesamt entsanden zusätzlich zu den bereits in Abschnitt 4.1.3 erwähnten noch folgende weitere PHP-Klassen zur Impementierung von Crowdsourcing-Kampagnen:

`Campaign`
: Repräsentiert eine Crowdsourcing-Kampagne, deren Definition als XML-Datei vorliegt. Bietet Methoden zur Berechnung von Ground-Truth-Score und Target-Score und der Belohnungspunkte.

`GroundTruthQuestion`
: Repräsentiert eine Ground-Truth-Frage mit einer laufenden Nummer, dem Text der Frage, der Liste der Antworten und dem Score.

`GroundTruthAnswer`

Repräsentiert eine Antwortmöglichkeit einer Ground-Truth-Frage mit einer laufenden Nummer, dem Text der Antwort und einem booleschen Wert, der anzeigt, ob die Antwort richtig oder falsch ist.

`Location`

Repräsentiert den Ort eines Teilnehmers mit Breiten- und Längengrad und Methoden zur Distanzberechnung und zur Verschleierung.

Auch die Klasse `Campaign` verwendet *SimpleXML* für den Zugriff auf den XML-Baum der Kampagnendatei. Initialisiert wird ein `Campaign`-Objekt mit der XML-Definition dieser Datei als Zeichenkette. Zur Ermittlung der entsprechenden Scores dienen die Methoden `getGroundTruthScore` und `getTargetScore`. Die Methode `getRewardPoints` gibt die gemäß der Berechnungsformel im Reward-Teil der Kampagnendefinition berechnete Anzahl an Belohnungspunkten zurück. Abbildung 4.1 zeigt ein Klassendiagramm der vier für die Kampagnendurchführung implementierten PHP-Klassen.

Abbildung 4.1: UML-Klassendiagramm aller für die Kampagnendurchführung implementierten PHP-Klassen

4.3 Integration in bestehenden SANE-Code

Die Kampagnenvalidierung wird unmittelbar beim Upload der Kampagnendefinition durchgeführt. Dazu dient die SANE-Methode `uploadCampaign`. Dies erfolgt in den geplanten zwei Schritten: zunächst die Validierung gegen die XML-Schemadefinition (siehe Abschnitt 4.1.2), dann die Validierung der zusätzlichen Constraints über die Klasse `CampaignValidator` (siehe Abschnitt 4.1.3). Die Mindestverschiebungsdistanz der Teilnehmer-Location, die unter anderem dem Konstruktor von `CampaignValidator` übergegeben wird, wurde über die Konfigurationsdatei des SANE auf 50 km festgelegt. Schlägt die Kampagnenvalidierung fehl, erfährt der HTTP-Client das über den Statuscode 415 (*Unsupported Media Type*) und ggf. eine Liste der Validierungsfehler.

4.3.1 Ablauf

Für die eigentliche Durchführung der Crowdsourcing-Kampagne wird jeder Methodenaufruf auf einem Crowdsourcing per HTTP abgefangen und nach dafür passenden Kampagnen gesucht (die Kampagnendefinitionen müssen vorher via `uploadCampaign` hochgeladen und dem SANE bekannt gemacht worden sein). Die Datei `cs_includer.php` ist hierfür die zentrale Anlaufstelle. Der implementierte Algorithmus durchläuft in etwa folgende Schritte:

1. Extrahiere aus dem HTTP-Request das angefragte Crowdsourcing und die angefragte Methode.
2. Für alle dem SANE bekannten Kampagnendefinitionen:
    - Wenn das angefragte Crowdsourcing und die angefragte Methode mit den Domänenangaben in der Kampagnendefinition übereinstimmen *und* die Kampagne zum aktuellen Zeitpunkt aktiv ist:
        - Hole anhand der im HTTP-Request mitgesendeten Device-ID die Benutzerdaten aus der Datenbank.
        - Ermittle alle zur Verfügung stehenden Diskriminatorwerte des Teilnehmers/der Contribution.
        - Ermittle anhand der Diskriminatorwerte den Target-Score des Teilnehmers.
        - Berechne anhand des Target-Scores die Anzahl der dem Teilnehmer zustehenden Belohnungspunkte gemäß der Kampagnendefinition und inkrementiere dementsprechend den in der Datenbank gespeicherten Wert für die Belohnungspunkte.
3. Rufe die eigentlich angefragte Crowdsourcing-Methode auf.

Auf die Implementierung eines Mechanismus für Ground-Truth-Fragen wurde aus Komplexitätsgründen noch verzichtet. Hierbei kommen einige zusätzliche Faktoren ins Spiel, die den Rahmen dieser Arbeit sprengen würden. Die Antworten des Teilnehmers auf eventuelle Ground-Truth-Fragen müssten zunächst gesondert gesammelt werden, um dann bei einem kampagnenrelevanten Methodenaufruf zur Auswertung zur Verfügung zu stehen.

4.3.2 Datenbankanpassung

Um die Teilnehmer-Location (bzw. die obfuskierte Version davon) zu erfassen, bietet sich eine SANE-globale Methode an, mit der ein Teilnehmer seinen momentanen Standort dem SANE bekanntmachen kann. Eine solche Methode existierte zum Zeitpunkt dieser Arbeit noch nicht. Es wurden allerdings in der Datenbank Felder zur Speicherung der letzten bekannten Location eines Benutzers angelegt, die für den Diskriminatorwert verwendet werden. Diese Felder sollen von der hypothetischen SANE-Methode jeweils aktualisiert werden. Abbildung 4.2 zeigt alle Änderungen am Datenbankschema, die zur Durchführung von Crowdsourcing-Kampagnen nötig waren, im Überblick.

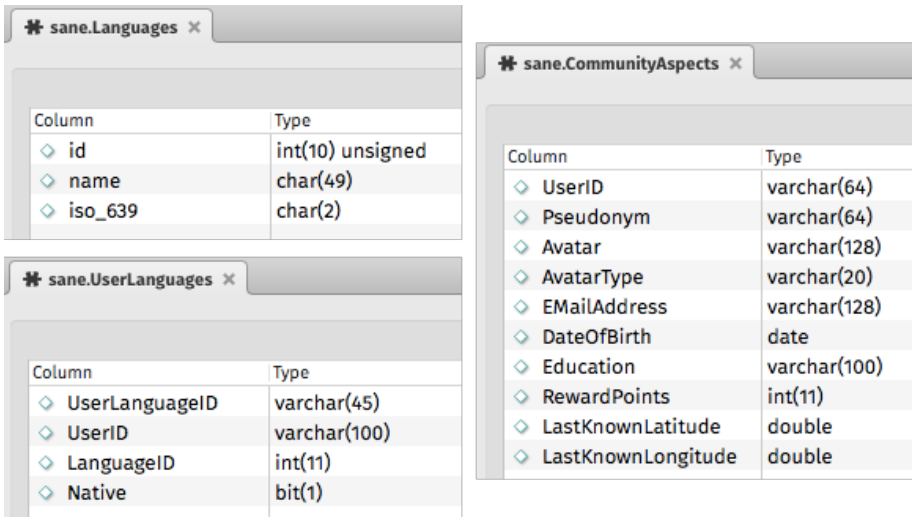

Abbildung 4.2: Übersicht über alle kampagnenrelevanten Änderungen am Datenbankschema

Die Tabelle `UserLanguages` bildet die n:m-Beziehung zwischen den Benutzern und deren gesprochenen Sprachen ab, zusammen mit der Information, ob es sich um die/eine Muttersprache handelt oder nicht. Die Sprachen selbst sind mit ihrem ISO-639-Code und ihrem Namen in der Tabelle `Languages` abgelegt.

4.4 Zusammenfassung

Im Rahmen der Implementierung wurde ein XML-Schema (XSD) für das in Kapitel 3 entworfene XML-Format zur Definition von Crowdsourcing-Kampagnen angelegt. Zur Anwendung kommt die Schemadefinition beim Upload einer Kampagnendatei auf eine SANE-Instanz. Der Upload schlägt fehl, wenn die Kampagnendatei nicht valide im Sinne des Schemas ist. Außerdem findet eine Validierung weiterer, in XSD nicht darstellbarer Bedingungen und Abhängigkeiten in der Kampagnendefinition im PHP-Code des SANE statt.

Vorausgesetzt, der Upload einer Kampagnendefinition verlief fehlerfrei, soll von nun an jeder Methodenaufruf innerhalb der Domäne der Kampagne (die Domäne setzt sich zusammen aus einem Crowdsourcing und einer Methode dieses Crowdsourcings) im Sinne der Kampagne

behandelt werden. Dazu gehört die Berechnung von Target- und Ground-Truth-Score, die auf Angaben von bzw. über den Teilnehmer beruhen sowie das Zuweisen von Belohnungspunkten, die in der Datenbank abgelegt werden sollen. Diese Geschäftslogik für Crowdsourcing-Kampagnen wurde ebenfalls im PHP-Code des SANE umgesetzt.

Zur Datenerfassung der Diskriminatorwerte eines Teilnehmers sowie der Anzahl seiner Belohnungspunkte wurde das Datenbankschema entsprechend erweitert. Die Diskriminatorwerte finden sich größtenteils in der Tabelle `CommunityAspects`. Zusätzlich hinzugekommen sind Tabellen zur Speicherung der von den Teilnehmern gesprochenen Sprachen.

Die Proof-Of-Concept-Implementierung für Crowdsourcing-Kampagnen ist damit umgesetzt worden. Eine Ausnahme stellt das Thema *Ground Truth* dar. Hierfür sind komplexere Eingriffe in den Ablauf beim Aufruf einer Crowdsourcing-Methode notwendig. Die Antworten eines Teilnehmers sollen prinzipiell von der eingesetzten Client-App erfasst und dem SANE bekannt gemacht werden, wo sie dann für die Auswertung des Ground-Truth-Scores benutzt werden können. Für die Berechnung der Belohnungspunkte eines Teilnehmers wurde im Rahmen dieser Arbeit der Ground-Truth-Score grundsätzlich auf 0 festgesetzt. Die korrekte Implementierung wird somit späteren Erweiterungen überlassen.

5 Evaluation

Nach der Implementierung der konzipierten Kampagnendefinition soll nun gezeigt werden, inwieweit diese für eine reale Durchführung einer Crowdsourcing-Kampagne geeignet ist. Außerdem soll die Fehlerfreiheit des geschriebenen Codes demonstriert werden. Dazu werden in diesem Kapitel die durchgeführten Tests zur Evaluation der Kampagnenlogik erläutert. Außerdem werden einige darüber hinausgehende Fragen zur Benutzbarkeit der hier vorgestellten Kampagnendefinition und des dadurch entstehenden Zusatzaufwands besprochen.

5.1 Unit-Tests und statistische Auswertung

Sämtlicher PHP-Code, der im Zuge der Implementierung entstanden ist, wurde mit dem Framework *PHPUnit*[1] in der Version 4.5.0 getestet. Dabei handelt es sich um Unit-Tests, also Tests, die einzelne Teile des Codes in Isolation auf ihr definiertes Verhalten prüfen.

5.1.1 Kampagnenvalidierung

Zum Testen der Kampagnenvalidierung wurden vier verschiedene, beispielhafte Kampagnendefinitionen entwickelt. Drei davon waren absolut korrekt, erfüllten also sowohl das per XSD vorgeschriebene Schema als auch die zusätzlichen Constraints (siehe Abschnitt 4.1.3) und nutzten dabei verschiedene Definitionsmöglichkeiten, z. B. verschiedene Score-Verteilungen oder Datums-/Zeitangaben. Die vierte Kampagnendefinition war zwar ebenfalls valide im Sinne der Schemadefinition, enthielt aber sieben Verstöße gegen die weiteren Constraints. Listing 5.1 zeigt den Test-Code für die Validierung dieser fehlerhaften Kampagnendefinition.

```
public function testValidity() {
    date_default_timezone_set('Europe/Berlin');
    $validator = new CampaignValidator(
        $_SERVER['PWD'] . '/tests/ex4.xml', 50);
    $result = $validator->checkValidity();

    $this->assertFalse($result->isValid());
    $this->assertEquals(count($result->getErrorList()), 7);
}
```

Listing 5.1: Unit-Test zur Validierung einer fehlerhaften Kampagnendefinition mit sieben Fehlern

[1] https://phpunit.de

Es wird zum einen getestet, ob die Validitätsprüfung überhaupt fehlschlägt, also die Methode `isValid` den Wert `false` zurückgibt. Zum anderen soll die Fehlerliste (Methode `getErrorList`) genau sieben Fehlermeldungen beinhalten. Die anderen drei Testfälle für die gültigen Kampagnendefinitionen testen dementsprechend, ob `isValid` jeweils `true` zurückliefert.

5.1.2 Kampagnenausführung

Auch die Kampagnenlogik selbst ist mit Unit-Tests abgedeckt. Hier gibt es im Wesentlichen zwei Testfallklassen, denen zwei verschiedene Kampagnendefinitionen zugrunde liegen. Getestet wird das Scoring der einzelnen Diskriminatoren sowie das gesamte Target-Scoring, das Ground-Truth-Scoring, die Berechnung der Belohnungspunkte, die Domänengültgkeit und die zeitliche Begrenzung der Kampagne. Außerdem gibt es Tests für die Distanzberechnung zwischen und die Verschleierung von Locations.

Target- und Ground-Truth-Scoring

Für das Scoring der einzelnen Diskriminatoren gibt es jeweils Tests, die sowohl verschiedene Werte innerhalb als auch außerhalb des jeweiligen Intervalls gültiger Diskriminatorwerte auf die korrekte Score-Vergabe testen. Dabei werden – sofern für den jeweiligen Diskriminator möglich – auch lineare und binäre Score-Verteilungen getestet. Darüber hinaus existiert ein Test für das Target-Gesamt-Scoring, also für die Methode `getTargetScore`, die intern die einzelnen Scoring-Methoden der Diskriminatoren aufruft.

Ähnlich war das Vorgehen beim Erstellen der Tests für das Ground-Truth-Scoring. Hier wurden verschiedene Antwortmuster der Teilnehmer auf die Ground-Truth-Fragen simuliert, wobei es aufgrund der Natur von Multiple-Mark-Questions (siehe Abschnitt 3.1.2) natürlich verschiedene Abstufungen von „richtig“ und „falsch“ geben kann. Auch eventuell in der Kampagnendefinition spezifizierte Score-Thresholds werden im Code berücksichtigt und dementsprechend auch getestet, sowohl bei der Ground Truth als auch beim Target.

Location-Verschleierung und -Distanzberechnung

Der Test für die Distanzberechnung ist denkbar simpel. Hier werden Orte mit bekannten räumlichen Abständen herangezogen und damit die Methode `distanceTo` der Klasse `Location` auf die korrekte Berechnung geprüft. Etwas komplizierter ist der Sachverhalt beim Test der Location-Verschleierung, da hier die Zufallskomponente dazukommt.

Auswertung der Verschiebungsrichtungen. Prinzipiell beinhaltet der Testfall für die Obfuskation eine wiederholte Ausführung der zu testenden Methode und anschließende statistische Auswertung der Ergebnisse. Zusätzlich wird bei jeder Iteration geprüft, ob der Abstand zur generierten obfuskierten Location den maximalen Verschiebungsradius nicht übersteigt. Nun reicht die pure Untersuchung der Verschiebungsdistanz nicht aus. Es soll auch sichergestellt werden, dass etwa gleich oft nach Norden und nach Süden bzw. nach Osten und nach Westen verschoben wird. Listing 5.2 zeigt den Codeausschnitt, mit dem

getestet wird, ob jeweils in beide Richtungen mindestens ein Drittel und höchstens zwei Drittel der Obfuskationen abgezielt haben.

```
// ...
for ($i = 0; $i < $iterations; $i++) {
    $loc_obf = $loc_real->getObfuscatedCopy($maxRadius);
    // ...
    $latLessCount += $lat_real - $loc_obf->getLatitude() > 0 ? 1 : 0;
    $lonLessCount += $lon_real - $loc_obf->getLongitude() > 0 ? 1 : 0;
}
// ...
$this->assertGreaterThanOrEqual($iterations / 3, $latLessCount);
$this->assertLessThanOrEqual($iterations * 2 / 3, $latLessCount);
$this->assertGreaterThanOrEqual($iterations / 3, $lonLessCount);
$this->assertLessThanOrEqual($iterations * 2 / 3, $lonLessCount);
```

Listing 5.2: PHPUnit-Code zum Test auf plausible Anzahl von Location-Obfuskationen in sämtliche Richtungen

Auswertung der Verschiebungsdistanzen. Die andere statistische Auswertung, die im Test-Code passiert, betrifft die Verschiebungsdistanzen selbst. Betrachtet man beispielsweise die Obfuskation einer Location Loc_{real} mit einem maximalen Verschiebungsradius von $m = 50\,\text{km}$ und zerlegt den dadurch entstehenden Umkreis um Loc_{real} in fünf Ringe der Breite $b = 10\,\text{km}$, so könnte man naiverweise annehmen, dass die Wahrscheinlichkeit, in welchem der Ringe die obfuskierte Location Loc_{obf} landet, sich nach dem Anteil des Flächeninhalts des Rings am Flächeninhalt des gesamten Kreises richtet. Diese Flächenanteile und das zugrunde liegende Prinzip der Obfuskation sind in Abbildung 5.1 visualisiert[2].

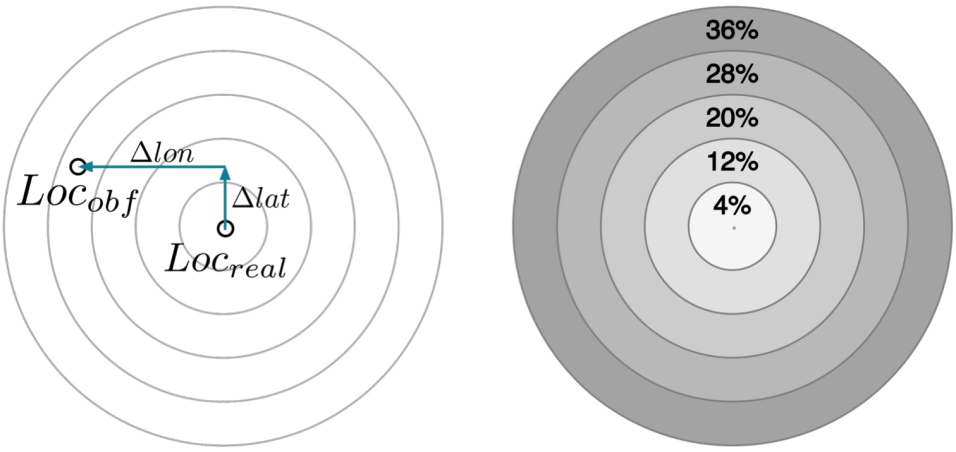

Abbildung 5.1: Prinzip der Location-Obfuskation mit separater Längen- und Breitengradverschiebung (l.) und jeweilige Anteile von fünf gleich breiten Kreisringen am Flächeninhalt des Gesamtkreises (r.)

[2]Die angegebenen Flächenanteile gelten sowohl für Kreise in der Ebene als auch für in dieser Arbeit relevante Kreise auf Kugeloberflächen.

Die Auswertung der tatsächlich vom Obfuskations-Algorithmus erzeugten Locations zeigt aber eine leicht abweichende Verteilung. Bei zwei exemplarischen Durchläufen des Test-Codes mit jeweils 1 000 000 Iterationen ergaben sich die relativen Häufigkeiten, die Abbildung 5.2 zeigt. Darin ist jeweils die Verteilung der obfuskierten Locations auf die Kreisringe dargestellt, einmal mit einem Verschiebungsradius von $m = 50\,\text{km}$ und einer Unterteilung in fünf Ringe (links) und einmal mit $m = 100\,\text{km}$ und einer Unterteilung in zehn Ringe.

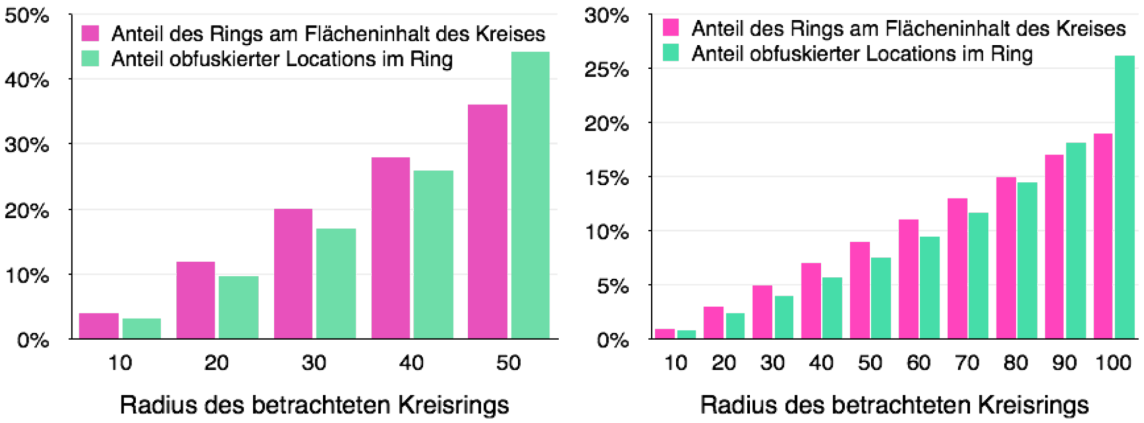

Abbildung 5.2: Verteilung von je 1 000 000 vom implementierten Algorithmus zufällig generierten, obfuskierten Locations auf fünf bzw. zehn Kreisringe bei Verschiebungsradien von $m = 50\,\text{km}$ (l.) und $m = 100\,\text{km}$ (r.)

Es fällt auf, dass überproportional viele der generierten Locations in den jeweils äußersten Ringen liegen. Die Ursache liegt in der Art, wie die Verschleierung abläuft: Zunächst wird die Verschiebung des Breitengrades zufällig gewählt, und zwar beliebig im Intervall zwischen 0 und dem Winkel, der eine Verschiebung um m bewirkt; erst danach erfolgt die Verschiebung des Längengrades. Dabei muss darauf geachtet werden, dass die entstehende Gesamtverschiebung den Radius m nicht überschreitet. Auf jeden Fall aber vergrößert diese zweite, separate Verschiebung den Abstand zwischen Loc_{real} und Loc_{obf} nochmals gegenüber der reinen Breitengradverschiebung.

Wären Breiten- und Längengradverschiebung komplett unabhängig aus demselben Intervall wählbar, würde sich die Verteilung der durch die Verschleierung entstehenden Abstände zwischen realer und obfuskierter Location nach dem zentralen Grenzwertsatz der Statistik einer Normalverteilung annähern. Im hier implementierten Algorithmus unterliegt die nachgestellte Längengradverschiebung aber wie besprochen der zusätzlichen Einschränkung, dass die zuvor gewählte Breitengradverschiebung berücksichtigt werden muss, damit beide zusammen eine Maximalverschiebung um m bewirken. Die Eigenschaften der Zufallsverteilung verschlechtern sich somit nochmals. Die idealerweise angestrebte Gleichverteilung der Verschiebungsdistanzen entlang des Verschiebungsradius wird daher durch den hier entwickelten Algorithmus verfehlt.

Um den Unterschied zu verdeutlichen, der bei den Ergebnissen des Algorithmus entsteht, wenn Δlon_{max} (maximale Längengradverschiebung) nicht dynamisch auf Basis von Δlat (tatsächliche Breitengradverschiebung) berechnet wird, sondern stattdessen – genauso wie die maximale Breitengradverschiebung Δlat_{max} – aus dem Quotienten von Verschiebungsradius m und dem Abstand der Längengrade am Äquator $d_{lon} = d_{lat} = 111{,}319\,49\,\text{km}$,

wurden nochmals jeweils 10 000 Iterationen mit beiden Varianten des Algorithmus und vier verschiedenen Ausgangs-Locations Loc_{real} durchgeführt. Die Häufigkeitsverteilungen der entstandenen Verschiebungsdistanzen sind in Abbildung 5.3 dargestellt. Die Kurvenverläufe zeigen jeweils die Anzahl der obfuskierten Locations Loc_{obf}, die durch die Ausführung des Obfuskations-Algorithmus *mit* (rot) und *ohne* (blau) dynamische Anpassung von Δlon_{max} in einer bestimmten Distanz von Loc_{real} entfernt (quantisiert auf 0,5 km breite Intervalle) generiert wurden. Als Verschiebungsradius wurde $m = 50\,\text{km}$ verwendet. Die graue Linie zeigt jeweils den idealerweise angestrebten Verlauf der Verteilungen basierend auf den besprochenen Flächeninhalten der Kreisringe.

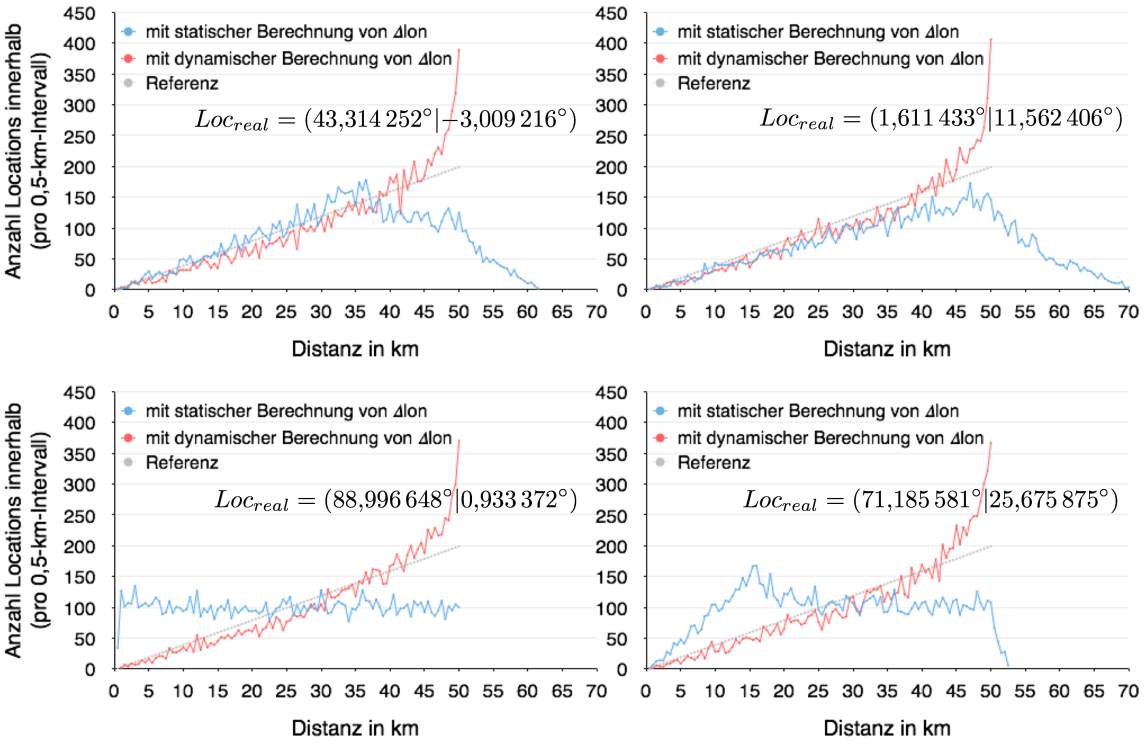

Abbildung 5.3: Verteilung der Obfuskations-Distanzen ($m = 50\,\text{km}$) bei 10 000 Durchläufen des Algorithmus mit vier verschiedenen Ausgangs-Locations und Berechnung jeweils mit (rot) und ohne (blau) Anpassung des Wertebereichs für Δlon

Unmittelbar fällt auf, dass die Verschiebungsdistanzen bei den berechneten Locations, die mit der Variation des Algorithmus ohne Anpassung von Δlon_{max} entstanden sind, größer als der verwendete Verschiebungsradius m (hier 50 km) sein können. Diese Erkenntnis ist trivial, denn schließlich hat die dynamische Anpassung genau den Sinn, die letztendlich erreichte Verschiebungsdistanz innerhalb des Intervalls $[0, m]$ zu halten. Interessanter ist der Verlauf der blauen Kurven in Abhängigkeit von der jeweiligen Ausgangs-Location Loc_{real} und die Tatsache, dass sich diese Abhängigkeit in den roten Kurven (die ja die Ergebnisse der in der Implementierung verwendeten Variation des Algorithmus widerspiegeln) nicht zeigt. In Bezug auf die blauen Kurven ändert sich die Verteilung je nach Breitengrad: Bei einer Location in der Nähe des Nordpols (Diagramm unten links in Abbildung 5.3) kommen beispielsweise alle Verschiebungsdistanzen zwischen 0 und m etwa gleich oft vor. Das liegt daran, dass das berechnete Δlon in Polnähe natürlich eine deutlich kürzere Wegstrecke ergibt als näher am Äquator und damit zur gesamten Verschiebungsdistanz

kaum etwas beisteuert. Das Resultat ist annähernd eine Gleichverteilung der Distanzen, da diese fast ausschließlich vom zufällig bestimmten Δlat abhängen. Dementsprechend steigt und fällt der Einfluss von Δlon auf die resultierende Verschiebungsdistanz mit der Wahl des Breitengrads von Loc_{real}, wenn – wie in den blauen Kurven – der Algorithmus *ohne* die dynamische Berechnung verwendet wird. Der Algorithmus *mit* der dynamischen Berechnung von Δlon_{max} (und damit auch Δlon) erzeugt aber Verteilungen von Verschiebungsdistanzen, die diese Anfälligkeit nicht besitzen. Tatsächlich erinnert der Verlauf aller roten Kurven in den Diagrammen an die Verteilungen aus Abbildung 5.2 und zeigt ebenfalls den starken Trend nach oben an den oberen Grenzen des Verschiebungsradius, der durch die vom Algorithmus erzwungenen Eigenschaften der Zufallsverteilung verursacht wird.

Insgesamt ist der Algorithmus für eine Proof-Of-Concept-Implementierung wie in dieser Arbeit aber vertretbar. Bei den 1 000 000 Iterationen im Beispiel aus Abbildung 5.2 mit Verschiebungsradius $m = 100\,\text{km}$ lagen etwa 37,3 % mehr Locations im äußersten Ring und etwa 21,9 % weniger Locations im innersten Ring als nach Gleichverteilung zu erwarten wäre. Auch in Abbildung 5.3 zeigt sich insgesamt, dass sich die Anzahl der obfuskierten Locations recht gut an die Referenzlinie annähert und nur jeweils im Grenzbereich stark abweicht. Die Gefahr liegt in der erhöhten Chance der Rekonstruierbarkeit der realen Location, wenn von potentiellen Angreifern derartige Annahmen über die Verteilungswahrscheinlichkeiten getroffen werden können. Hier besteht Verbesserungsspielraum für zukünftige Erweiterungen dieses Ansatzes zur Implementierung von Crowdsourcing-Kampagnen.

Verbesserungsvorschläge. Das Grundproblem besteht darin, dass Längengrad- und Breitengradverschiebung getrennt voneinander zufällig bestimmt und dann überlagert werden. Eine optimale Lösung bestimmt per Zufall direkt die Verschiebungsdistanz r im Intervall $[0, m]$ sowie einen Verschiebungswinkel ϕ, sodass aus diesen Polarkoordinaten (r, ϕ) dann im nächsten Schritt die Geo-Koordinaten von Loc_{obf} berechnet werden können. Auch hier sollte bestenfalls die sphärische Geometrie berücksichtigt werden, da die Verscheibungsradien durchaus groß sein können und der Unterschied zu einer Berechnung mit ebener Geometrie dann signifikant wäre.

5.2 Anwendungsfall

Um die Kampagnenimplementierung an einem konkreten Anwendungsfall testen zu können, wurde ein Beispiel-Crowdsourcing namens `CampaignTest` mit nur einer Methode `contribute` erstellt. Die Methode tut nichts weiter, als die Zeichenkette `Hello world` zurückzugeben. Zusätzlich wurde eine Kampagne definiert, die folgende Target-Anforderungen an die Teilnehmer stellt:

- Das Alter soll zwischen 20 und 30 Jahren liegen. Die Score-Verteilung ist dabei linear mit dem Zentralwert bei einem Alter von 25 Jahren, einem Minimal-Score von 1 und einem Maximal-Score von 8.
- Die Location soll innerhalb von 200 km um den Geo-Punkt $(41{,}299\,133° | -1{,}552\,866°)$ liegen und ggf. einen Score von 4 generieren. Die Score-Verteilung ist binär.

- Falls der höchste Bildungsabschluss die Promotion oder Habilitation ist, sollen dafür 8 Score-Punkte vergeben werden, sonst 0.
- Sollte der Teilnehmer nativ Spanisch sprechen, erhält er dafür 6 Score-Punkte, bei Spanisch als Zweitsprache noch 3.
- Die Anzahl der Belohnungspunkte soll mindestens 10 betragen, um einen Minimal-Score von 2 zu generieren. Für 20 Belohnungspunkte wird ein Score von 3 vergeben und zur Berechnung der Scores für alle anderen Diskriminatorwerte wird entlang dieser Referenzpunkte linear interpoliert.

Darüber hinaus wurde für das Target ein Score-Schwellwert von 10 definiert, der von Teilnehmern mindestens erreicht werden soll. Außerdem ethält die Kampagnendefinition eine Formel zur Berechnung der Belohnungspunkte, die den vom Teilnehmer erzielten Target-Score mit 1,5 multipliziert.

Anschließend wurde in der Datenbank ein User hinterlegt, der die aufgelisteten Anforderungen teilweise erfüllt. Tabelle 5.1 listet auf, welche Diskriminatorwerte dieser Testnutzer aufweist und welche Target-Scores sich daraus ergeben.

Diskriminator	**Diskriminatorwert**	**Score**
Alter	25	8
Location	(39,784 453°|−1,121 246°) (Distanz zu Target-Location etwa 172,5 km)	4
Bildung/Ausbildung	Abitur	0
Sprachen	Walisisch (nativ), Sesotho, Moldawisch	0
Belohnungspunkte	8	0
	Gesamt	**12**

Tabelle 5.1: Erfüllung der Target-Kriterien eines fiktiven Testnutzers und daraus resultierende Scores gemäß der zu testenden Kampagnendefinition

Mit diesen Diskriminatorwerten erzielt der Nutzer einen höheren Target-Score (12), als es der Threshold (10) verlangt. Dementsprechend sollte dieser Score verwendet werden, um mithilfe der im Reward-Teil der Kampagnendefinition spezifizierten Formel die Anzahl der zu vergebenden Belohnungspunkte auf $12 \cdot 1{,}5 = 18$ zu berechnen. Diese Belohnungspunkte sollen dann zum bisherigen Punktestand in der Datenbank (8) addiert werden, sodass sich ein neuer Punktestand von $18 + 8 = 26$ ergibt. Dieser Anwendungstestfall wurde – ebenso wie die Unit-Tests aus Abschnitt 5.1 – erfolgreich von der Implementierung erfüllt.

5.3 Kommunikations-Overhead durch Kampagnen

In dem Moment, wo zu einem Crowdsourcing eine Kampagne definiert werden soll, entsteht zwangsläufig zusätzlicher Traffic durch den Crowdsourcing-Anbieter, der die Kampagnendatei auf eine SANE-Instanz hochladen muss. Außerdem soll evtl. eine Interaktion mit

den Crowdsourcing-Teilnehmern erfolgen, die ebenfalls Platz und Bandbreite beansprucht. Diese Anforderungen sollen in diesem Abschnitt untersucht werden.

5.3.1 Mindestanforderungen für Kampagnen-Upload

Aufgrund des XML-Formats, das für die Definition einer Crowdsourcing-Kampagne in dieser Arbeit entworfen wurde, lässt sich eine kleinstmögliche Dateigröße angeben, mit der eine gültige Kampagnendefinition erzielt wird. Diese Definition gibt weder ein `ground-truth`- noch ein `target`-Element an, muss dann aber ein `reward`-Element beinhalten. Darin muss es wiederum ein `formula`-Element geben, in dem die Berechnungsformel für die Belohnungspunkte nichts weiter als eine hartkodierte, einstellige Zahl ist. Als Domänenangabe verwendet die minimale Kampagnendefinition die doppelte Wildcard `/*/*`, das Namespace-Kürzel darf nur einen Buchstaben lang sein und auch die XML-Präambel muss weggelassen werden. Listing 5.3 zeigt diese kleinstmögliche Kampagnendefinition in voller Länge von 127 Zeichen und damit einem Speicherbedarf (bei Kodierung in UTF-8) von 127 Bytes.

```
<c:campaign xmlns:c="http://rn.inf.tu-dresden.de/SANE" domain="/*/*"><c:re
ward><c:formula>0</c:formula></c:reward></c:campaign>
```

Listing 5.3: Kleinstmögliche, gültige Crowdsourcing-Kampagnendefinition in 127 Zeichen (Zeilenumbruch hier nur aus Platzgründen eingefügt)

Solch eine Kampagnendefinition wäre vermutlich wenig sinnvoll, denn sie würde jedem Teilnehmer jedes Crowdsourcings bedingungslos eine feste Zahl an Belohnungspunkten (hier 0, aber auch jede andere, einstellige Zahl verändert die Dateigröße nicht) gutschreiben. An dieser Stelle sollen die ermittelten 127 Bytes aber als absolute Untergrenze für jede Kampagnendefinition herhalten. Nach oben ist die Größe der Kampagnendefinition natürlich theoretisch unbegrenzt, aber die einzigen Möglichkeiten, wie sie wirklich exzessiv werden kann, sind übermäßig lange Ground-Truth-Fragen (bzw. -Antworten) oder eine übermäßig lange Berechnungsformel für Belohnungspunkte (was eher unwahrscheinlich ist). Alle anderen Elemente der Definition – wovon es ohnehin nur wenige gibt – bieten lediglich Platz für einzelne Zahlen oder wohldefinierte Zeichenketten, sodass die Gesamtgröße der Datei sich in aller Regel maximal im einstelligen Kilobyte-Bereich abspielen sollte. Der einmalige Upload einer Definitionsdatei einer Crowdsourcing-Kampagne wird also nach menschlichem Ermessen keinen nennenswerten Traffic verursachen, auch wenn dazu wiederum der Download der Schemadefinition (9 668 Bytes) zum Zwecke der Validierung notwendig ist.

5.3.2 Benachrichtigungen über Kampagnenteilnahme

Ein weiterer Aspekt, der bisher in dieser Arbeit ausgespart wurde, ist die zusätzliche Kommunikation, die durch die Benachrichtigung des Teilnehmers über die Kampagne anfällt. Ob solche Benachrichtigungen von Nutzern gewünscht sind und wie sie aus Sicht der Benutzerinteraktion abgewickelt werden könnten, wird genauer in Abschnitt 5.4 erörtert. Eine simple Bestätigung/Benachrichtigung über die Zahl der vergebenen Belohnungspunkte innerhalb der HTTP-Response auf den entsprechenden Methodenaufruf dürfte aber kaum ins Gewicht fallen. Der wesentlich größere Overhead entsteht durch HTTP selbst, denn

bei jeder Response werden aufs Neue wieder zahlreiche Header mitgeschickt. Listing 5.4 zeigt die vollständige HTTP-Response, die beim Aufruf der Methode `contribute` des zur Evaluation erstellten Crowdsourcings `CampaignTest` entsteht.

```
HTTP/1.1 200 OK
Server: Apache/2.4.10 (Unix) OpenSSL/1.0.1i PHP/5.5.15 mod_perl/2.0.8-dev
Perl/v5.16.3
X-Powered-By: PHP/5.5.15
Content-Type: text/plain
Signature: 123456
Date: Fri, 17 Apr 2015 17:36:52 GMT
Content-Length: 11
Timestamp: 2015-04-17T17:36:52Z
Connection: close

Hello world
```

Listing 5.4: HTTP-Response beim Aufruf der Methode `contribute` des Crowdsourcings `CampaignTest` mit einer Länge von 288 Bytes, davon 275 Bytes allein für HTTP-Headers (ein zusätzlicher Zeilenumbruch wurde hier aus Platzgründen eingefügt)

Nur die allerletzte Zeile (`Hello world`) gehört zum eigentlichen Payload der Response. Der deutlich größere Teil der Response (275 Bytes) wird durch die HTTP-Header verursacht. Selbst wenn im Payload nun noch eine Information über die Teilnahme an der Kampagne bzw. die erhaltenen Belohnungspunkte untergebracht werden würde, dürfte deren Größe im Vergleich zu den bei jeder Response anfallenden Header-Daten vernachlässigbar sein. Sollte die durch diese Benachrichtigungen entstehende Zusatzkommunikation wider Erwarten doch Platzprobleme verursachen, kann immernoch eine Lösung implementiert werden, in der sie nicht bei jeder Response, sondern stattdessen gebündelt (beispielsweise einmal pro Tag) dem Teilnehmer gesendet wird. Sie wäre damit auch vom eigentlichen Crowdsourcing-Traffic entkoppelt und würde diesen so in keiner Weise mehr beeinträchtigen.

5.4 Probandenbefragung

Nachdem die Konzeption der Kampagnen aus Sicht eines Crowdsourcing-Anbieters erfolgreich abgeschlossen und in die bestehende Architektur von SANE überführt wurde, sollen nun auch noch die Wünsche möglicher Crowdsourcing-Teilnehmer erfasst werden. Zur Evaluation in Bezug auf Benutzbarkeit und die Erwartungshaltungen, mit denen potentielle Nutzer einem solchen System begegnen, wurde daher eine Probandenbefragung durchgeführt. Die Befragung war mündlich und fand unter acht Teilnehmern (drei weiblich, fünf männlich) im Alter von 23 bis 49 Jahren (Median 29, Durchschnitt 30,9) jeweils in Einzelgesprächen statt. Die Rücklaufquote betrug 100%. Das Konzept einer Crowdsourcing-Kampagne musste den Teilnehmern zunächst erklärt werden. Bereits den Begriff „Crowdsourcing“ kannten nur drei der acht Befragten wirklich, eine weitere Person stellte zumindest eine Verbindung mit dem Wort „Crowdfunding“ her, das ihr geläufig war. Drei der Probanden sind oder waren in ihrem Leben zu irgendeinem Zeitpunkt in eine informatische Hochschulausbildung involviert, an einem Crowdsourcing aktiv teilgenommen hatte nur ein einziger. Der vollständige

Fragebogen, mit dem die Teilnehmer konfrontiert wurden, umfasste folgende Fragen bzw. Aufgaben:

1. Beschreibe möglichst vollständig den chronologischen Ablauf, den du von einer Crowdsourcing-Kampagne aus Benutzersicht erwartest.
2. Falls aus bisherigen Ausführungen nicht ersichtlich: An welchen Stellen erwartest du Rückmeldungen, Benachrichtigungen oder sonstige Interaktionen von oder mit dem System?
3. Falls aus bisherigen Ausführungen nicht ersichtlich: Wodurch würdest du persönlich dich zur Teilnahme an einem Crowdsourcing motiviert fühlen?
4. Bei welchen der folgenden Informationen über dich wärst du bereit, diese für die Zwecke einer Crowdsourcing-Kampagne preiszugeben und bei welchen nicht?
    - Geburtsdatum
    - Location
    - höchster Bildungsabschluss
    - gesprochene Sprachen

Mit der Beschreibung des erwarteten Ablaufs einer Crowdsourcing-Kampagne taten sich die Befragten durchweg schwer, sodass viele gezielte Nachfragen nötig waren. Viele Teilnehmer drifteten in ihren Ausführungen in Bereiche ab, die zwar das Crowdsourcing allgemein betrafen, aber wenig Verwertbares in Bezug auf die Durchführung von Kampagnen, wie in dieser Arbeit diskutiert, lieferten. Fast alle kamen von sich aus auf die Motivation zur Teilnahme zu sprechen und es kristallisierte sich heraus, dass für die meisten Befragten (sechs von acht) nur dann eine Teilnahme an einem Crowdsourcing zu erwarten ist, wenn sie von den darin generierten Ergebnissen direkt profitieren. Für zwei Probanden darf dieser Profit ausschließlich pekuniärer Natur sein, damit eine Mitarbeit infrage kommt. Eine Person brachte die Idee auf, Teilnehmer des Crowdsourcings mit materiellen Gimmicks (Kugelschreiber, Schlüsselanhänger o. Ä.) zu belohnen. Nachfragen zum logischen und zeitlichen Ablauf einer Crowdsourcing-Kampagne ergaben, dass die Teilnehmer der Befragung eine Information über die Kampagne im Vorfeld für selbstverständlich halten. Das gilt auch für die Bestätigung der Teilnahme an der Kampagne im Nachhinein. Die Erwähnung der Möglichkeit einer „zufälligen“ und unaufgeforderten Teilnahme stieß bei den meisten Befragten eher auf Verwunderung. Hinsichtlich der Beantwortung der Ground-Truth-Fragen äußerten drei Personen den Wunsch, dies im Zuge der Ersteinrichtung bzw. Registrierung in der letztendlich verwendeten Client-Anwendung ein für alle Mal erledigen zu können, da sie diesen Vorgang für potentiell störend hielten.

Darüber hinaus wurde jeder Proband zu seinen möglichen Bedenken gegenüber der Preisgabe persönlicher Informationen zur Erfassung seiner Diskriminatorwerte befragt. Die Ergebnisse sind in Tabelle 5.2 kompakt dargestellt. Unproblematisch waren bei allen die Speicherung des Geburtsdatums zur Berechnung des Alters sowie der gesprochenen Sprachen. Jeweils zwei der Befragten gaben an, nur ungern ihren höchsten Bildungsabschluss oder ihren geographischen Ort bekanntzugeben.

Darauf hingewiesen, dass letzterer in der aktuellen Konzeption nur obfuskiert gespeichert wird, erklärte einer der beiden Bedenkenträger, dass es für ihn eher auf die generelle Ver-

Information	Von acht Befragten fanden die Herausgabe ... ... problematisch	... unproblematisch
Geburtsdatum	0	8
Location	2	6
Bildungsabschluss	2	6
Sprachen	0	8

Tabelle 5.2: Ergebnisse der Befragung nach den Bedenken bezüglich der Preisgabe von persönlichen Informationen

trauenswürdigkeit des Crowdsourcing-Anbieters ankäme, die sich beispielsweise auch in Erfahrungsberichten anderer Benutzer zeigt, und weniger auf derartige Implementierungsdetails. Der andere äußerte nach diesem Hinweis leicht erhöhte Bereitschaft, seine Location zu Zwecken der Kampagne zur Verfügung zu stellen. Einig waren sich alle Befragten darin, dass ein späterer Widerruf der Kampagnenteilnahme möglich sein soll.

5.5 Zusammenfassung

In umfassenden Unit-Tests wurde zunächst nach bestem Wissen und Gewissen die Fehlerfreiheit des implementierten Codes sichergestellt. In Bezug auf die Location-Verschleierung wurden statistische Auswertungen unternommen, die eine Schwäche im verwendeten Obfuskations-Algorithmus aufdeckten. Dies zeigt sich darin, dass überproportional viele obfuskierte Locations mit großen Distanzen zur realen Location generiert werden. Die Ursache liegt in den schlechten Eigenschaften der Zufallsverteilung, die durch die Funktionsweise des Algorithmus begünstigt werden (Näheres in Abschnitt 5.1.2). Schließlich wurde die korrekte Funktionalität der Kampagnenimplementierung mit einem Anwendungstestfall überprüft. Dazu wurde ein Test-Crowdsourcing sowie eine darauf definierte Testkampagne angelegt und anschließend getestet, ob ein Methodenaufruf durch einen für die Kampagne geeigneten Teilnehmer diesem tatsächlich die erwartete Zahl an Belohnungspunkten gutschreibt.

Auch der durch Crowdsourcing-Kampagnen verursachte Overhead in der HTTP-Kommunikation zwischen Client-Anwendungen und dem SANE wurde untersucht. Dieser kann allerdings sowohl was den Upload der Kampagnendefinition, als auch was die Kommunikation von Kampagnendetails mit den Teilnehmern angeht, für die allermeisten zu erwartenden Anwendungen vernachlässigt werden. Dennoch kann im Notfall problemlos eine Lösung impementiert werden, bei der Teilnehmer ihre Kampagneninformationen beispielsweise nur täglich in gebündelter Form erhalten, um Traffic zu sparen.

Eine Befragung unter acht Personen sollte Anforderungen von bisher mit der Thematik nicht befassten, potentiellen Nutzern eines Systems wie dem hier konzipierten sammeln. Es stellte sich heraus, dass es den Befragten schwerfiel, sich eine genaue Vorstellung vom Ablauf einer Crowdsourcing-Kampagne zu machen. Was sich sowohl aus den expliziten Aussagen der Teilnehmer als auch aus den impliziten Erwartungshaltungen, die in vielen ihrer Ausführungen deutlich wurden, ableiten lässt, ist die Notwendigkeit einer Aufforderung zur Kampagnenteilnahme bzw. zumindest die Information über das Vorhandensein einer solchen.

Da auch viele der Befragten erwähnten, dass sie von der Teilnahme am Crowdsourcing eine Belohnung erwarten, sollte eine in der Praxis verwendete Umsetzung die Nutzer ausdrücklich darüber informieren, dass eine Gutschrift von Belohnungspunkten stattgefunden hat und ob und in welcher Form diese möglicherweise in materielle Werte übertragbar ist. Hinsichtlich des Privatsphärebedürfnis äußerten sich je zwei Teilnehmer der Befragung verhalten bis widerwillig gegenüber der Preisgabe von höchstem Bildungsabschluss und ihrem aktuellen Aufenthaltsort. Auch der Hinweis über die nur verschleiert erfolgende Speicherung der Location schien die beiden Personen nicht wirklich umzustimmen. Allerdings lohnt sich die Obfuskation der Teinehmer-Location dennoch höchstwahrscheinlich, da sie keine große Komplexität erfordert und möglicherweise von vielen Nutzern gar nicht vorhergesehene Angriffsvektoren ausschaltet. Außerdem ist die Stichprobe der Befragten (acht) bei weitem nicht groß genug, um repräsentative Aussagen über die Anforderungen und Wünsche aller Benutzer zu machen.

6 Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurde eine Möglichkeit entwickelt, um Crowdsourcing-Kampagnen zu definieren. Diese Kampagnen können für ein in der Crowdsourcing-Architektur SANE bereits bestehendes Crowdsourcing festgelegt werden. Die wesentlichen Konzepte dieser Arbeit werden in diesem Kapitel zusammengefasst und anschließend ein Ausblick auf mögliche Erweiterungen gegeben.

6.1 Definition von Crowdsourcing-Kampagnen

Eine Crowdsourcing-Kampagne wird vom Crowdsourcing-Anbieter in einer XML-Datei definiert und umfasst drei wesentliche Bestandteile:

- **Target:** Legt mithilfe von Diskriminatoren fest, welche Eigenschaften Teilnehmer der Crowdsourcings idealerweise haben sollten, um für die Kampagne wertvoll zu sein und quantifiziert dies durch den *Target-Score*.
- **Ground Truth:** Definiert eine Reihe von *Multiple-Mark-Questions*[1], deren Beantwortung durch einen Teilnehmer Aufschluss über dessen Eignung für die Crowdsourcing-Aufgabe liefert und quantifiziert die Qualität der Antworten durch den *Ground-Truth-Score*.
- **Reward:** Gibt mithilfe einer Formel an, wie aus dem Target-Score und dem Ground-Truth-Score eine Anzahl von Belohnungspunkten für den Teilnehmer berechnet werden soll.

Die Diskriminatoren, die im Target benutzt werden können, sind das Alter, die Location, der höchste Bildungsabschluss, die Anzahl der Belohnungspunkte und die gesprochenen Sprachen des Teilnehmers sowie der Zeitpunkt seiner Contribution. Für die Teilnehmer-Location wurde zusätzlich aus Datenschutzgründen ein Obfuskationsalgorithmus entwickelt, sodass der Ort nur verschleiert gespeichert wird. In der Kampagnendefinition kann dann angegeben werden, für welche Diskriminatorwerte welcher Score vergeben werden soll. Die einzelnen Diskriminator-Scores ergeben dann zusammen den Target-Score. Für die Score-Verteilung pro Diskriminator gibt es grundsätzlich zwei Möglichkeiten: Diese können als *binär* oder *linear* definiert werden. Binäre Score-Verteilungen schütten den spezifizierten Score je nach Diskriminatorwert entweder komplett oder gar nicht aus, bei linearen Verteilungen wird zwischen einem Minimal- und einem Maximal-Score interpoliert. Dementsprechend können lineare Score-Verteilungen nur bei numerisch darstellbaren Diskriminatorwerten definiert werden.

[1] Fragen, bei denen aus einer vorgegebenen Liste von Antwortmöglichkeiten eine oder mehrere richtige Antworten ausgewählt werden sollen

Um den Ground-Truth-Score aus den Antworten des Teilnehmers auf die Ground-Truth-Fragen zu berechnen, wurde die Berechnungsvoschrift von Tarasowa und Auer [TA13] verwendet, die eine Technik zum Scoring bei Multiple-Mark-Questions beschreibt. Die Grundidee besteht darin, eine Basispunktzahl für das Ausschöpfen der richtigen Antworten zu geben und davon eine Strafpunktzahl für ausgewählte falsche Antworten abzuziehen.

6.2 Implementierung

Der erste Schritt zur Implementierung des Konzepts war das Anlegen einer Schemadefinition in XSD, gegen die die XML-Kampagnendefinitionen validiert werden. Zudem wurden alle nicht per XSD darstellbaren Einschränkungen und Regeln im PHP-Code umgesetzt, der beim Upload der Definitionsdatei auf den SANE ausgeführt wird. Nach erfolgreichem Upload und erfolgreicher Validierung sollen anschließend bei jedem Methodenaufruf auf einem im SANE registrierten Crowdsourcing die Spezifikationen der Kampagne berücksichtigt werden. Zunächst wurde das Datenbankschema erweitert, um die Diskriminatorwerte des Teilnehmers erfassen zu können. Diese werden dann ggf. ausgelesen, der Target-Score gemäß der Kampagne berechnet und die daraus berechnete Zahl an Belohnungspunkten zum alten Punktestand des Teilnehmers addiert.

Aus der Implementierung ausgeklammert wurde die Durchführung der Ground-Truth-Questions. Diese können zwar vollständig in der Kampagnendefinition angelegt werden und auch der Code zur Berechnung der Scores basierend auf den Antworten ist vorhanden, allerdings müsste, damit tatsächlichen Crowdsourcing-Teilnehmern die Fragen gestellt und ihre Antworten erfasst werden können, der konzeptionelle Ablauf und das Kommunikationsprotokoll bei einem Crowdsourcing im Sinne von SANE zunächst erweitert werden.

6.3 Evaluationsergebnisse

Für die Evaluation des implementierten Codes wurden Unit-Tests geschrieben sowie ein Anwendungstestfall entworfen und durchgeführt. Um die Funktionalität der Location-Verschleierung zu prüfen, erfolgten eine Reihe von statistischen Auswertungen, die sowohl die Verteilung der Richtungen als auch der Distanzen der Verschiebung untersuchten. Dabei zeigte sich in Bezug auf die Distanzen eine Schwäche des verwendeten Algorithmus, die zu überproportional vielen obfuskierten Locations nahe der maximalen Verschleierungsentfernung führt. Hier besteht noch Verbesserungsbedarf für spätere Weiterentwicklungen.

Außerdem wurde die Frage nach dem zusätzlichen Kommunikationsbedarf, der durch Crowdsourcing-Kampagnen entsteht, diskutiert. Es stellte sich heraus, dass weder der Kampagnen-Upload noch die Kommunikation irgendwelcher Kampagnendetails während oder nach der Ausführung hier sonderlich ins Gewicht fallen dürfte. Der größte Overhead in der Kommunikation entsteht durch HTTP selbst, und hier insbesondere durch die zahlreichen Header-Informationen, die bei jeder Response zum Client geschickt werden.

Abschließend erfolgte eine Probandenbefragung unter acht Personen zur Erfassung der Vorstellungen und Wünsche von potentiellen Teilnehmern eines Crowdsourcings bzw. einer Crowdsourcing-Kampagne. Viele der Befragten taten sich schwer mit der Beschreibung

ihrer Erwartungen an ein solches System, jedoch kann aus den Ergebnissen der Befragung insgesamt der Schluss gezogen werden, dass die Nutzer in irgendeiner Weise über das Stattfinden und ihre Teilnahme an einer Kampagne informiert werden sollten. Hinsichtlich des Datenschutzbedürfnisses zeigten sich bei je zwei der Befragten Bedenken gegenüber der Veröffentlichung ihrer aktuellen Location und ihres höchsten Bildungsabschlusses.

6.4 Ausblick

In Zukunft können Crowdsourcing-Kampagnen auf einer SANE-Instanz durchgeführt werden. Es fehlt noch eine sinnvolle Implementierung für Ground-Truth-Fragen, damit alle in dieser Arbeit vorgestellten Konzepte vollständig umgesetzt sind. Zusätzlich zur reinen Funktionalität der Kampagne muss dafür gesorgt werden, dass teilnahmewillige Nutzer über das Vorhandensein einer Kampagne aufgeklärt und im Nachhinein über die erfolgte Belohnung informiert werden.

In der aktuellen Umsetzung wird darauf vertraut, dass der Crowdsourcing-Anbieter, der die Kampagne erstellt, dabei ein möglichst sinnvolles Target angibt. Es hängt also alleine von dessen Einschätzung ab, welche Diskriminatoren verwendet und auf welche Diskriminatorwerte abgezielt werden soll. Eine Weiterentwicklung in diesem Bereich könnte beispielsweise an die in Kapitel 2 vorgestellte Arbeit von Li et. al. [LZF14] angelehnt sein und das für eine Kampagne optimale Target automatisiert bestimmen lassen. Das vom Crowdsourcing-Anbieter definitierte Target würde eine solche Implementierung dann nur noch als grobe Startvorgabe interpretieren und Mechanismen wie evolutionäre Algorithmen verwenden, um eine Art „selbstlernendes“ bzw. „selbstoptimierendes“ Kampagnen-Target zu erzielen.

Generell können neue, bisher nicht verwendete Diskriminatoren eingesetzt werden. Pro neuem Diskriminator muss eine Möglichkeit zur Speicherung geschaffen und ein neues XML-Element in der XSD-Datei definiert werden. Eventuell müssen auch neue Arten von Score-Verteilungen neben den hier konzipierten linearen und binären Verteilungen erdacht werden. Denkbar ist unter anderem die Vergabe des Scores auf Basis von Wertebereichen, wie z. B. IP-Adressbereichen. Dazu müssten Selektoren definiert werden können, mit denen die gewünschten Wertebereiche auswählbar wären. Solche Selektoren könnten wiederum auch Zeichenketten nach Mustern filtern etc.

Insgesamt ist das Target wahrscheinlich das entscheidenste Element für den Erfolg und die Effizienz einer Crowdsourcing-Kampagne. Um herauszufinden, welche der hier angesprochenen möglichen Erweiterungen die erfolgversprechendsten und welche zusätzlichen Diskriminatoren sinnvoll sind, bieten sich Benutzerstudien an, in denen unter Realbedingungen ein Crowdsourcing mit verschiedenen Kampagnendefinitionen eingesetzt wird und die Ergebnisse verglichen werden.

Literaturverzeichnis

[Ahn06] Luis von Ahn. Games with a Purpose. *Computer*, 39(6):92–94, June 2006.

[AvdE11] Oguz Ali Acar and Jan van den Ende. Motivation, Reward Size and Contribution in Idea Crowdsourcing. *DIME-DRUID ACADEMY. Comwell Rebild Bakker, Aalborg, Denmark*, 2011.

[BAT02] C. Daniel Batson, Nadia Ahmad, and Jo–Ann Tsang. Four Motives for Community Involvement. *Journal of Social Issues*, 58(3):429–445, 2002.

[DRH11] Anhai Doan, Raghu Ramakrishnan, and Alon Y. Halevy. Crowdsourcing Systems on the World-Wide Web. *Commun. ACM*, 54(4):86–96, April 2011.

[DSN+11] Sebastian Deterding, Miguel Sicart, Lennart Nacke, Kenton O'Hara, and Dan Dixon. Gamification. Using Game Design Elements in Non-gaming Contexts. In *CHI '11 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '11, pages 2425–2428, New York, NY, USA, 2011. ACM.

[EAGLDG12] Enrique Estellés-Arolas and Fernando González-Ladrón-De-Guevara. Towards an Integrated Crowdsourcing Definition. *J. Inf. Sci.*, 38(2):189–200, April 2012.

[Har11] Christopher Harris. You're hired! An Examination of Crowdsourcing Incentive Models in Human Resource Tasks. In *Proceedings of the Workshop on Crowdsourcing for Search and Data Mining (CSDM) at the Fourth ACM International Conference on Web Search and Data Mining (WSDM)*, pages 15–18, 2011.

[HJV13] Chien-Ju Ho, Shahin Jabbari, and Jennifer Wortman Vaughan. Adaptive Task Assignment for Crowdsourced Classification. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 534–542. JMLR Workshop and Conference Proceedings, 2013.

[HSBS13] Tenshi Hara, Thomas Springer, Gerd Bombach, and Alexander Schill. Decentralised Approach for a Reusable Crowdsourcing Platform Utilising Standard Web Servers. In *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*, UbiComp '13 Adjunct, pages 1063–1074, New York, NY, USA, 2013. ACM.

[IG14] Panagiotis G. Ipeirotis and Evgeniy Gabrilovich. Quizz: Targeted Crowdsourcing with a Billion (Potential) Users. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 143–154, New York, NY, USA, 2014. ACM.

[Int67] International Organization for Standardization (ISO). Language codes - ISO 639, 1967. http://www.iso.org/iso/home/standards/language_codes.htm, abgerufen am 17.01.2015.

[LZF14] Hongwei Li, Bo Zhao, and Ariel Fuxman. The Wisdom of Minority: Discovering and Targeting the Right Group of Workers for Crowdsourcing. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 165–176, New York, NY, USA, 2014. ACM.

[Nie14] Nielsen Holdings N.V. Lean back/lean forward, 2014. https://definedterm.com/lean_back_lean_forward, abgerufen am 26.04.2015.

[ope14] openthesaurus.de, 2014. https://www.openthesaurus.de/synonyme/Kampagne, abgerufen am 02.12.2014.

[RPH+12] Dana Rotman, Jenny Preece, Jen Hammock, Kezee Procita, Derek Hansen, Cynthia Parr, Darcy Lewis, and David Jacobs. Dynamic Changes in Motivation in Collaborative Citizen-science Projects. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, pages 217–226, New York, NY, USA, 2012. ACM.

[TA13] Darya Tarasowa and Sören Auer. Balanced Scoring Method for Multiple-Mark Questions. In *5th International Conference on Computer Supported Education (CSEDU 2013)*, 2013.

[The15] The PHP Group. PHP: Basics – Manual, 2001-2015. http://php.net/manual/en/language.variables.basics.php, abgerufen am 05.03.2015.

[TX08] Christian Terwiesch and Yi Xu. Innovation Contests, Open Innovation, and Multiagent Problem Solving. *Management Science*, 54(9):1529–1543, 2008.

Abbildungsverzeichnis

Listings

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Diplomarbeit mit dem Titel „Definition von Crowdsourcing-Kampagnen“ selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie alle wörtlich oder sinngemäß übernommenen Gedanken und Zitate als solche kenntlich gemacht habe. Ich erkläre ferner, dass ich die vorliegende Arbeit an keiner anderen Stelle als Prüfungsarbeit eingereicht habe oder einreichen werde.

Dresden, 15.05.2015