Lastenheft

„Praxis der Software-Entwicklung“
Wintersemester 2015/16

„Crowd Control –
Entwicklung von Steuerungsmechanismen
für das Crowd Computing“

Lehrstuhl IPD Böhm

Motivation

Die Idee von Crowd-Computing ist, die „kollektive Intelligenz“ realer Personen zur Lösung von Aufgaben einzusetzen, die nicht automatisch lösbar sind. Solche Aufgaben kann man auf einer Crowd-Plattform wie Amazon Mechanical Turk (mTurk) veröffentlichen. Dort bearbeiten sie Menschen, sogenannte Crowd-Worker. Sie erhalten im Gegenzug vom Aufgabensteller eine monetäre Entlohnung, in vielen Fällen in Einklang mit dem Umfang der Aufgabe nur wenige Cents. Das Anwendungsspektrum von Crowd-Computing ist breit und reicht von kleinen und einfachen Aufgaben wie der Annotation von Bildern bis hin zu komplexen Aufgaben wie dem Schreiben ganzer Artikel.

Das Ziel unserer Forschung ist es herauszufinden, ob und unter welchen Umständen Crowd-Worker zur Generierung von kreativen Inhalten eingesetzt werden können. Aufgaben können dabei beispielhaft das Schreiben einer Pointe für eine Sitcom, gegeben den Anfang eines Dialogs zwischen beispielsweise Sheldon und Penny, oder das Kreieren eines neuen Fotowitzes sein, gegeben ein Foto von zwei Politikern. Allerdings kann der Aufgabensteller sich nicht sicher sein, dass die Crowd-Worker auch Inhalt in der gewünschten Qualität liefern, und dies ist in unseren bisherigen Untersuchungen auch tatsächlich oft nicht der Fall. Bei kreativen Inhalten ist eine objektive Bewertung schwierig bis unmöglich. Wir sehen deshalb keine Alternative dazu, die Bewertung ebenfalls durch Crowd-Worker vornehmen zu lassen.

Als Teil unserer bisherigen Arbeit wurde eine konfigurierbare Software („CreativeCrowd“) entwickelt, die im Rahmen eines Experiments bei mTurk mehrere kreative Antworten sowie entsprechende Bewertungen einsammelt und die Bezahlung der Worker, die gemäß vorgegebener Regeln entlohnt werden, selbständig vornimmt. Als ‚Experiment‘ bezeichnen wir den gesamten Ablauf, der für die Bearbeitung einer Aufgabe wie „Erstelle einen Fotowitz zu Foto X.“ erforderlich ist und das Einsammeln mehrerer Kreativantworten, entsprechender Bewertungen und Bezahlung der Worker, wenn die entsprechenden Bedingungen erfüllt sind, umfasst.

In unseren bisherigen Untersuchungen haben sich insbesondere die folgenden Probleme als schwerwiegend herausgestellt:

(1) Betrugsversuche (Spoof bzw. offensichtlich lieblos generierte Inhalte), aber auch Inhalt, der unbrauchbar ist, obwohl die Worker sich anscheinend bemüht haben, sind zahlreich.

(2) Bei der Verwendung von mTurk hat man keinen Einfluss auf die Auswahl der Worker für bestimmte Aufgaben. (Man kann lediglich einzelne Worker für alle

Aufgaben sperren, die man dort publiziert.) Aus Forschungssicht wäre genau das jedoch sehr wünschenswert.

Aufgabe

Ihre Aufgabe besteht darin, für die bestehende Infrastruktur (mTurk und CreativeCrowd insbesondere) Lösungen zu entwickeln, die eine verbesserte Steuerung/Kontrolle der Worker ermöglichen, wie folgt:

a) Das erste o. g. Problem lässt sich zumindest abmildern, wenn Aufgabensteller die Möglichkeit haben, ihre Aufgabenstellung während eines Experiments noch anzupassen/zu verfeinern. Es geht insbesondere darum, nachträglich noch spezifizieren zu können, wie die kreativen Antworten nicht beschaffen sein sollen. Denn erfahrungsgemäß lässt sich nicht alles vorhersehen, sondern manche Anforderungen werden erst offensichtlich, wenn kreative Antworten mit ungewünschten Eigenschaften vorliegen. Das Erstellen von Aufgabenstellungen, insbesondere indem man bereits bestehende Aufgaben anpasst, soll möglichst komfortabel sein. Anders ausgedrückt geht es in diesem Aufgabenteil um die Erstellung einer geeigneten Benutzeroberfläche für Aufgabensteller.

b) Das zweite Problem wollen wir angehen, indem Aufgaben nicht nur bei mTurk, sondern auch auf anderen Crowd-Plattformen – mit anderen Regeln – bearbeitet werden. Beispielhaft sollen Sie eine Kopplung von CreativeCrowd mit einer Open-Source Plattform für Mikrotasks entwerfen und realisieren. Mit derartigen Plattformen können Experimentatoren dann selbst Populationen von Individuen/Workern zusammenstellen (bzw. eigene Populationen mit der mTurk-Workforce in kontrollierter Weise mischen). Im Folgenden sprechen wir der Einfachheit halber von pyBossa; dies beinhaltet aber keine Festlegung auf diese Technologie, und Sie können gern auch etwas anderes verwenden.

Da sich mTurk auf logischer Ebene von anderen Plattformen unterscheidet, ist der Entwurf der angedachten Kopplung nicht trivial. Der Zusammenhang zu (a) besteht insbesondere darin, dass die Spezifikation der Aufgabenstellung Vorgaben enthalten darf, welche Worker was genau tun dürfen.

Weitere Erläuterungen

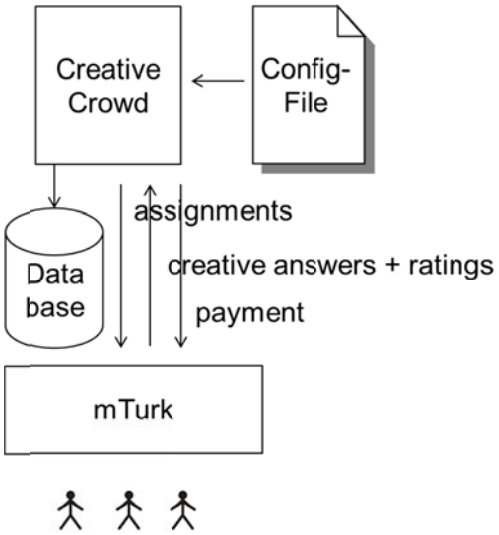
Die angestrebte Anbindung der existierenden Infrastruktur an pyBossa ist aus den folgenden zwei Gründen nützlich/sinnvoll:

(21) „Lückenbüßer-Funktionalität“. (Wurde oben noch nicht gesagt, wird gleich erklärt.)
(22) allgemeinere Möglichkeiten für Experimente. (Wurde oben bereits genannt und wird gleich noch einmal erklärt.)

(21) bedeutet Folgendes: In unseren bisherigen Experimenten war es des Öfteren der Fall, dass es für eingereichte Kreativbeiträge nicht genügend Bewertungen gab, und das Experiment deshalb nicht vernünftig abgeschlossen werden konnte. In diesem Fall würden wir die ausstehenden Tasks gern einem, sagen wir, *Personenkreis unter unserer Kontrolle* übertragen wollen, z. B. den Mitarbeitern des Lehrstuhls. Die Idee ist also, dass diese Personen genau die pyBossa-Community/die Gruppe von selbst rekrutierten Workern bilden, von der oben die Rede war.

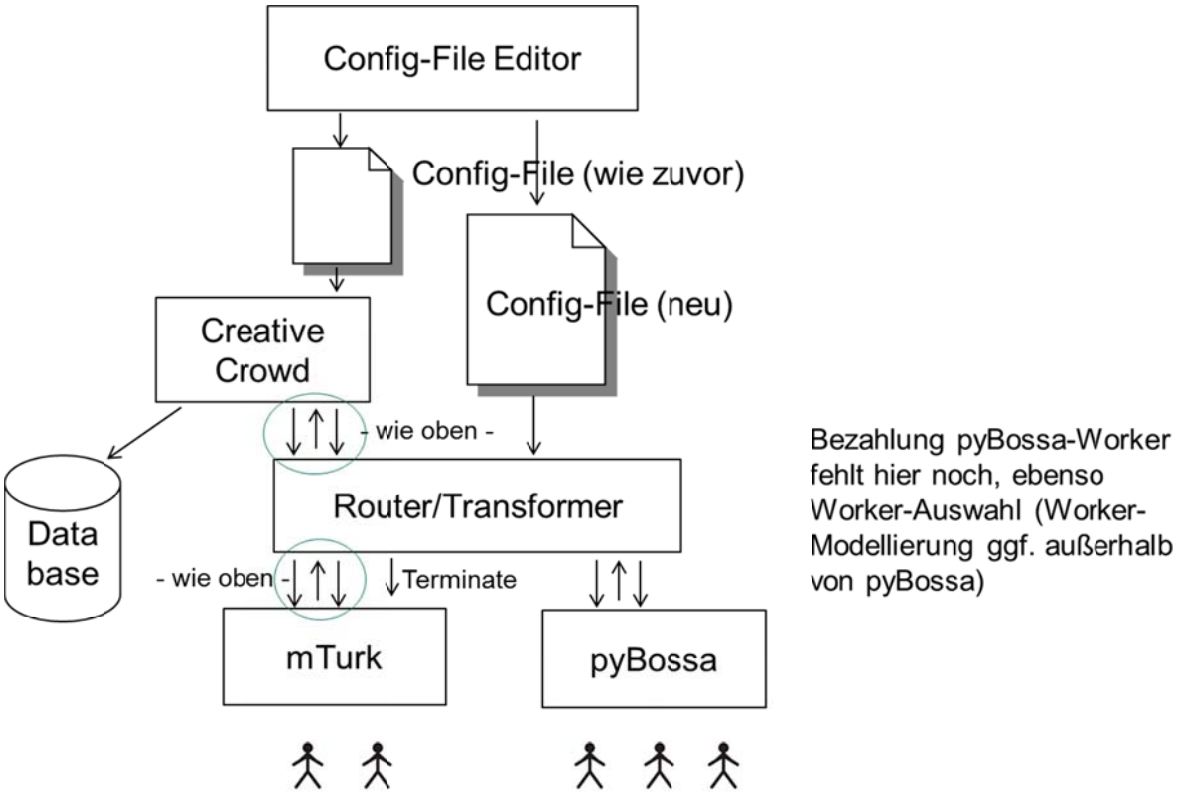
Mit (22) geht es uns darum, dass wir gern Crowdworking-Experimente mit ausgewählten Personen durchführen möchten. Seien es komplette Experimente ausschließlich mit ausgewählten Personen, sei es, dass wir sicherstellen wollen, dass eine Experiment-Population einen bestimmten Mindestanteil von Personen mit bestimmten Eigenschaften enthält, sei es, dass nur ausgewählte Personen bestimmte Tasks ausführen (z. B. die allgemeine mTurk-Population darf Kreativbeiträge erstellen, die Bewertung erfolgt aber ausschließlich durch ‚unsere‘ Population – oder umgekehrt).

Auch wenn es, sagen wir, unüblich ist, Ihnen eine Software-Architektur vorzuschlagen, ist dies im vorliegenden Fall vermutlich für das Verständnis hilfreich, insbesondere weil bereits eine Software existiert, um die herum die Lösung entwickelt werden soll.



Die Abbildung oben zeigt die Architektur des derzeitigen Systems. ‚mTurk‘ ist die Amazon-Marktplattform für Mikrotasks. Die vom letztjährigen PSE-Team erstellte

Software ‚CreativeCrowd' schickt ihr Aufgaben, die im Rahmen eines Experiments zu bearbeiten sind, insbesondere Kreativaufgaben oder Bewertungsaufgaben oder Kombinationen davon, nimmt das Ergebnis entgegen und stößt ggf. Bezahlungen an. Die Aufgabenstellung eines Experiments und seine Parameter (z. B. wieviele Kreativantworten wollen wir, wie hoch ist die Bezahlung für eine gelungene Kreativantwort) sind Inhalt eines Config-Files, dessen Aufbau vorgegeben ist, und das ein Aufgabensteller bisher von Hand erstellen muss. Eine Datenbank enthält die Information zum Zustand eines laufenden Experiments.



Die Abbildung oben zeigt eine mögliche Architektur des angestrebten Systems. (Es kann sein, dass die Bearbeitung der Aufgabe erkennen lässt, dass eine andere Architektur besser ist; in diesem Fall soll natürlich diese bessere Variante umgesetzt werden – die Abbildung enthält ‚nur' den aktuellen Stand unserer Überlegungen.) Die bestehende Software CreativeCrowd funktioniert anscheinend weitgehend zuverlässig und soll nach Möglichkeit nicht modifiziert werden, daraus ergibt sich die vorgeschlagene Architektur weitgehend. Zu realisieren sind im Wesentlichen zwei Komponenten, der ConfigFile-Editor und die Router/Transformer-Komponente. Der Router/Transformer fängt die bisherigen mTurk-Aufrufe der CreativeCrowd-Komponente ab und übersetzt sie ggf. in pyBossa-Aufrufe, je nach Experimentaufbau. Analog übersetzt sie die Ergebnisse, die von pyBossa zurückkommen, ins bisherige Format. Es

kann grundsätzlich auch sein, dass der Router/Transformer mTurk-Aufrufe verändert. Eine Konfig-Datei (deren Aufbau Sie festlegen sollen) spezifiziert das Router/Transformer-Verhalten. Der angestrebte KonfigFile-Editor generiert nicht nur die Konfig-Datei für CreativeCrowd, sondern auch die für den Router/Transformer. – In der Abbildung nicht explizit enthalten ist die Bezahlung der pyBossa-Worker, ebenso die Möglichkeit, dass der Experimentator einen bestimmten Teil der pyBossa-Community für sein Experiment auswählen kann.

Wir sind jetzt in der Lage, die o. g. Punkte (21) und (22) ausführlicher zu erläutern. (21), die „Lückenbüßer-Variante“, d. h. wenn von mTurk nicht mehr genug Antworten kommen, sollen die verbleibenden Aufgaben der pyBossa-Community gestellt werden, ist ein MUSS-Kriterium, dessen Umsetzung nicht übermäßig kompliziert sein sollte (sieht man von der Transformation in möglicherweise andere Aufrufe und der Rücktransformation der Ergebnisse ab). Hinsichtlich (22), der Möglichkeit ausgefeilterer Experimente, unterscheiden wir zwischen den folgenden Fällen:

aa) Keine Unterscheidung zwischen den beiden Populationen (d. h. die mTurk-Worker einerseits und die pyBossa-Worker andererseits) in einem Experiment. (Gemeint ist, keine strukturelle Unterscheidung – es soll aber möglich sein, dass die Bezahlung von pyBossa-Workern anders ist als die der mTurk-Worker.)

bb) Differenzierung zwischen diesen Populationen, z. B.

- Kreativantworten sollen ausschließlich von mTurk-Workern, Bewertungen ausschließlich von pyBossa-Workern kommen.
- Kreativantworten sollen ausschließlich von mTurk-Workern kommen, Bewertungen können aus beiden Populationen kommen.

(Feinere Differenzierungen sind mMn nicht erforderlich, bzw. sehen wir nicht, was für ausgefeiltere Differenzierungen wirklich interessant sein könnten.)

Die Einbindung von pyBossa-Workern erlaubt außerdem die folgende, zur obigen Differenzierung orthogonale Unterscheidung:

aaa) Alle Mitglieder der pyBossa-Community dürfen an einem bestimmten Experiment teilnehmen.

bbb) Nur eine Teilmenge dieser Community soll mitmachen dürfen. Zwei Arten von Auswahlkriterien sind denkbar, nämlich

- ‚strukturelle‘ Eigenschaften der Worker, z. B. Alter oder Wohnort, also Eigenschaften, die wir vorher abgefragt haben. (Gehen Sie bitte der Einfachheit halber davon aus, dass die Angaben richtig sind, die ein Worker zu sich gemacht hat.)
- Verhaltensbasierte Eigenschaften. Z. B. die durchschnittliche Bewertung der Kreativantworten des jeweiligen Workers, die Anzahl der Bewertungen, die er bisher abgegeben hat usw. (Gesucht sind einfache, naheliegende

Kriterien – die Bearbeiter mögen sich welche überlegen und die dann umsetzen.)

Konfig-Editor – MUSS-Kriterien

<Um die folgenden Punkte komplett zu verstehen, ist es sinnvoll, dass Sie sich zunächst Config-Dateien von Experimenten ansehen, die bereits stattgefunden haben.>

- Laden existierender Konfigs und Speichern unter beliebigem Namen; Möglichkeit, geladene Konfigs beliebig zu editieren.
- ‚Redundanzfreiheit' der Eingabe. (In unseren Config-Dateien kommt Text i. Allg. mehrmals vor, z. B. der gleiche Text in der Beschreibung der Kreativaufgabe sowie in der Beschreibung der Bewertungsaufgabe. Mit Ihrem Editor soll der Aufgabensteller diesen Text nur einmal eingeben müssen.)
- Explizite Unterstützung unterschiedlicher Aufgabentypen. Der Aufgabensteller soll zunächst gefragt werden, von welchem Typ seine Aufgabe ist, und dann nur Eingaben machen können, die für diesen Aufgabentyp erforderlich sind. Aufgabentypen, mit denen wir bereits experimentiert haben, sind die folgenden:
  - Finden von Fotos im Internet zu einem bestimmten Thema gemäß unserer Vorgaben. Beispielsweise haben wir nach Fotos gefragt, die ‚Privatheit' in angenehmer Weise illustrieren, also kein Vorhängeschloss, Stacheldrahtzaun oder Ähnliches, (eine Aufgabe dieses Typs) oder die ‚Vertrauen' illustrieren (eine andere Aufgabe dieses Typs). Der Aufgabensteller soll hier die Möglichkeit haben, zum Zweck der Veranschaulichung URLs von Bildern beispielhaft anzugeben, die er akzeptieren würde, und solchen, für die das nicht der Fall ist.
  - Formulieren eines ‚Mean Tweets' zu einem Prominenten. (Wir zeigen Ihnen hierzu gern Beispiele.)

  Die Eingabe von Aufgaben eines Typs, den Sie nicht explizit modelliert haben, muss aber natürlich auch möglich sein.
- Text zur Bezahlung in Aufgabenstellung aus Parametereinstellungen automatisch generieren.

Transformer/Router – MUSS-Kriterien

- Einbindung von pyBossa (oder einer anderen OpenSource Crowdworker-Software Ihrer Wahl) mit der gleichen Funktionalität wie die existierende mTurk-Anbindung an CreativeCrowd. (Z. B. Anzahl Kreativantworten pro Experiment, Anzahl Bewertungen, Bewertungsoptionen, Kontrollfragen.)
- Möglichkeit der Bezahlung (ist meines Wissens derzeit nicht Teil von pyBossa) schaffen. (Es genügt das Schicken einer E-Mail mit einem Amazon-Geschenkgutschein. Bitte nicht mehrere E-Mails mit jeweils Cent-Beträgen.)
- Realisierung der o. g. ‚Lückenbüßer-Funktionalität'.

- Realisierung von (22), wie oben beschrieben. (Das ist möglicherweise umfangreich. Mir ist derzeit nicht klar, inwieweit pyBossa die Nutzer modelliert und ihr vergangenes Verhalten erfasst. Sie sollen sich überlegen, ob es sinnvoll ist, diesen Punkt mit einer pyBossa-Erweiterung oder einer neuen, separaten Komponente zu lösen, und die überlegene Lösung umsetzen.)

MUSS-Kriterien nichtfunktionaler Art

- Nutzbarkeit. Ein Außenstehender muss in der Lage sein, Ihre Software ohne Einweisung zu nutzen.
- Erweiterbarkeit. Da wir Ihre Software in zukünftigen Projekten nutzen wollen, soll sie übersichtlich und modular realisiert sein.
- Gute Dokumentation. Ist unabdingbar.
- Zuverlässigkeit. Es müssen mehrere ‚echte' Experimente stattfinden, die mehrere Tage dauern; Ihre Software darf nicht abstürzen o. ä. Sollte ein Absturz doch geschehen, muss es möglich sein, das Experiment in diesem Zustand fortzusetzen.