PPLib: towards systematic crowd process design using recombination and auto-experimentation

De Boer, Patrick ; Bernstein, Abraham

PPLib: Towards Systematic Crowd Process Design using Recombination and Auto-Experimentation

PATRICK M. DE BOER, University of Zurich
ABRAHAM BERNSTEIN, University of Zurich

1. INTRODUCTION

The field of crowd sourcing is growing quickly and strives to go past it's early stigma of being used only for very simple tasks such as image labeling and object tagging. Research in crowd processes focused intensely on quality assurance in crowd environments and yielded various patterns for structuring crowd workers such as Find-Fix-Verify [Bernstein et al. 2010], Iterative Refinement [Little et al. 2010] or Iterative Dual-Pathway [Liem and Chen 2011] – but when designing new applications for crowd sourcing, there is currently no guidance on how to decide which pattern to use or related questions such as the amount of crowd workers to be employed for a specific step in a pattern. *System designers are essentially left to pick and implement one of the patterns proposed before based purely on their intuition and without any assurance that it is a suitable or even a good choice.*

We propose to solve this problem by systematically exploring the design-space of complex crowd-sourced tasks via automated recombination and auto-experimentation for an issue at hand. Specifically, we propose an approach to finding the optimal process for a given problem by defining the deep structure of the problem in terms of it's abstract operators, generating all possible alternatives via the (re-)combination of the abstract deep structure with concrete implementations from a Process Repository, and then establishing the best alternative via auto-experimentation.

Since the Process Repository is made up of process fragments that build upon other fragments or complex processes, we go one step further than to simply try all previously proposed patterns for a given problem: our methodology allows us to generate (i.e. recombine) completely novel processes and apply them to any issue that has been formalized well enough.

To evaluate our methodology, we have implemented PPLib (pronounced "People Lib") – a program library that allows for such an automated recombination of known processes stored in an easily extensible Process Repository.

2. THE PPLIB METHODOLOGY

The PPLib Methodology is based on insights in the area of business process reengineering (BPR), where [Malone et al. 2003] proposed the use of the Process Handbook, a process repository encompassing more than 5000 business processes that were organized according to principles of coordination theory [Malone and Crowston 1994] as well as a specialization hierarchy [Battle et al. 2005]. The Process Recombinator [Bernstein et al. 1999] expanded on this repository and proposed to automatically recombine fragments of processes in order to generate novel business processes.
The first step to using the Recombinator was to identify the process deep structure, i.e. the main abstract processing steps that make up the core activities of the process.
The Recombinator would then recombine different abstract process elements of the process repository based on the specialization hierarchy. We applied the principles of the Recombinator to the domain of crowd sourcing, where we can further automatically evaluate the processes resulting from recombination through the use of human computation portals such as Amazon's Mechanical Turk[1].

[1] http://www.mturk.com

We have established the PPLib Process Repository (PPR) for crowd sourcing based on the same mechanics as the MIT process handbook. This allows us to explore the full design-space of crowd processes through the recombination of (nested-) process fragments within the repository and evaluate their performance when applied to a pre-specified process deep structure using auto-experimentation. We, hence, reduce the problem of designing crowd processes to defining such a process deep structure.

Recombination is based on a specialization hierarchy, where we implemented the base types for *create* and *decide* processes as outlined in [Malone et al. 2010] – with *create* and *decide* both being specializations of a *Human Computation Task*. Abstract crowd processes are stored in the Process repository, specializing one of the base types, and are capable of using parameters of any type – including *create* / *decide*-type parameters, which allows for processes to be nested.
When applied to a specific scenario, one defines the deep structure of the process and includes *Human Computation Tasks*, where one would like to employ recombination. Some parameters, such as worker instructions or desired worker counts, may be specified in the deep structure itself to prune the design-space to be explored. The Recombinator then finds all processes specializing a given type in the PPR and then processes all parameters of this specialization as follows: (i) for parameters asking for a subprocess of specified type, it recursively finds allowed specializations of that type and processes them (ii) for simple parameter type, such as bounded integers (e.g. worker count) or worker instructions, it returns all allowed values. Each parameter combination yields an individual recombined process that can then be automatically evaluated in parallel. Through a user-defined utility function, the auto experimentation engine is then able to create a ranking of crowd processes for a given problem. In order to ensure the stability of a solution, one can use standard sampling methods to acquire a sufficiently large dataset of results per recombined crowd process.

3. THE PPLIB PROGRAMMING LIBRARY

The PPLib Programming Library is based on recent advances to develop program code that coordinates Humans and Computers alike. There are numerous noteworthy examples for existing domain specific languages, programming libraries and development environments [Barowy et al. 2012; Little et al. 2010; Kittur et al. 2011; Ahmad et al. 2011]. Some of them incorporate basic measures of quality assurance, e.g. Automan that uses a mechanism to sample answers to the same single choice question until a certain confidence value is reached. However, none of them aid a developer with the *systematic creation of complex crowd processes*, which is the main strength of PPLib.
We built our programming library on top of Scala, allowing a programmer to reuse his familiar development toolset to formalize the deep structure of a problem. PPLib is human computation portal agnostic and has support for Amazon Mechanical Turk as well as CrowdFlower built-in, which allows a developer to integrate crowd workers of both platforms in a crowd process, essentially transcending the boundaries between crowd portals. In order to support long-running transactions, PPlib supports the Crash&Rerun pattern introduced by [Little 2011].
Our process repository currently contains 15 crowd processes, out of which 5 are concrete, i.e. they do not nest any other processes, hence serving as basic building blocks for other processes. We have made it very easy to integrate new processes and process fragments into the repository for anyone. New processes can be published onto a public repository by independent developers and imported by developers around the world into their PPLib application using SBT dependency management[2] (Ivy[3] and Maven[4]).

[2] http://www.scala-sbt.org/
[3] http://ant.apache.org/ivy/
[4] http://maven.apache.org/

4. EVALUATION

By the time of submission, PPLib was used to process more than 18'900 micro tasks, involving more than 650 unique workers (by Turker ID).

4.1 Text translation

The first evaluation of PPLib focused on translating a German article from the news portal of the Swiss bank UBS to English[5]. Since the article was available in both, English and German, we have used the English version as a benchmark for our later evaluation. We have identified the process deep structure for text translation to consist of (i) the machine translation of the article from German to English using Google Translate, (ii) then disassembling the resulting baseline translation into its individual sentences that are then (iii) refined by crowd workers in terms of grammar, followed by (iv) constructing the translated article by ordering the refined sentences according to their position in the original article. The third step designates a *Human Computation task (create)*, for which the Recombination Engine generated 41 different implementations that were then executed on Amazon's Mechanical Turk. The 41 resulting translations were rated by 10 qualified experts, each in terms of Fluency and Adequacy.
Through recombination, we generated numerous processes that have not been officially proposed before and established the best-performing process in the translation environment through auto-experimentation and evaluated it using the 10 experts. Our 5 best processes performed very well in terms of adequacy and fluency, with one of them even exceeding the benchmark in the adequacy dimension. We contend that systematically exploring the design-space of possible processes yields better results than picking among previously proposed patterns, since the process performing best in the context of text translation has not been proposed before.

4.2 Text shortening

The second evaluation of PPLib was about shortening text passages, where we used an article from the news portal *the verge* with 1444 characters[6]. The process deep structure was comparable to the one used in section 4.1, except that we did not include a machine translation step and configured the Recombinator to use different instructions for crowd workers. Due to reusing the same type of *Human Computation Task* (*create*), the Recombinator generated the same set of base crowd processes, i.e. 41 crow processes, among them variants of the well-known Find-Fix-Verify pattern (FFV). We executed these processes 4 times and found that 5 automatically created processes reliably performed better in terms of text length than the best variant of FFV (Mann-Whitney test, $P < 0.0001$) – while still having very high scores in terms of adequacy. The best FFV process performed well and was ranked 6th.

4.3 Differences in text translation and text shortening

The usage of the same abstract processes in both contexts enabled us to compare the performance of each generated process in either situation, with the goal of proving our initial claim that crowd processes depend on the environment they are applied in and that there is therefore no silver bullet in structuring crowd workers. The different performance of the generated processes between the two environments were confirmed by a Kendall-Tau rank correlation of 0.04 with $P=0.82$ for the complete absence of a relationship between the performance of the same crowd process in the two different contexts. We, hence, conclude, that *there indeed is no silver bullet for crowd process design, which emphasizes the need for a way to systematically engineer crowd processes for which recombination constitutes a first step*.

[5] http://www.ubs.com/global/en/about_ubs/about_us/news/news.html/en/2014/12/29/consumption-indicator.html
[6] http://www.theverge.com/2015/1/8/7517361/google-getting-ready-to-sell-auto-insurance-and-maybe-buy-coverhound

5. BIBLIOGRAPHY

Salman Ahmad, Alexis Battle, Zahan Malkani, and Sepander Kamvar. 2011. The Jabberwocky Programming Environment for Structured Social Computing. *Architecture* (2011), 53. DOI:http://dx.doi.org/10.1145/2047196.2047203

Daniel Barowy, Charlie Curtsinger, Emery Berger, and Andrew McGregor. 2012. AutoMan: A platform for integrating human-based and digital computation. *Proc. ACM Int. Conf. Object oriented Program. Syst. Lang. Appl. - OOPSLA '12* (2012), 639. DOI:http://dx.doi.org/10.1145/2384616.2384663

S. Battle, A. Bernstein, H. Boley, and B. Grosof. 2005. Semantic web services language (SWSL). *W3C Memb.* (2005), 1–61.

Abraham Bernstein, Mark Klein, and Thomas W. Malone. 1999. The process recombinator: a tool for generating new business process ideas. In *ICIS '99 Proceedings of the 20th international conference on Information*. 178–192.

Michael S. Bernstein et al. 2010. Soylent : A Word Processor with a Crowd Inside. (2010), 313–322.

Aniket Kittur, Boris Smus, Susheel Khamkar, and Robert E. Kraut. 2011. CrowdForge: Crowdsourcing Complex Work. In *Proceedings of the 24th annual ACM symposium on User interface software and technology - UIST '11*. 43–52. DOI:http://dx.doi.org/10.1145/2047196.2047202

Beatrice Liem and Yiling Chen. 2011. An Iterative Dual Pathway Structure for Speech-to-Text Transcription. In 37–42.

Greg Little. 2011. *Programming with Human Computation*. Massachusetts Institute of Technology.

Greg Little, Lydia B. Chilton, Max Goldman, and Robert C. Miller. 2010. TurKit: Human Computation Algorithms on Mechanical Turk. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology - UIST '10*. 57. DOI:http://dx.doi.org/10.1145/1866029.1866040

Thomas W. Malone and Kevin Crowston. 1994. The interdisciplinary study of coordination. *ACM Comput. Surv.* 26 (1994), 87–119. DOI:http://dx.doi.org/http://doi.acm.org/10.1145/174666.174668

Thomas W. Malone, Kevin Crowston, and George A. Herman. 2003. *Organizing Business Knowledge: the MIT Process Handbook*,

Thomas W. Malone, Robert Laubacher, and Chrysanthos Dellarocas. 2010. The collective intelligence genome. *IEEE Eng. Manag. Rev.* 38 (2010), 38. DOI:http://dx.doi.org/10.1109/EMR.2010.5559142