State of the Art in Crowdsourcing

Richard Bayerle
Technical University of Vienna
Studentnr. 1025259
e1025259@
student.tuwien.ac.at

Peter Klein
Technical University of Vienna
Studentnr. 8251105
e8251105@
student.tuwien.ac.at

Alexander Kumbeiz
Technical University of Vienna
Studentnr. 1228689
e1228689@
student.tuwien.ac.at

ABSTRACT

This paper provides an overview in the state of the art in crowd sourcing applications. A classification of crowd sourcing systems is shown according to the collaboration provided by users and the tasks performed by them. Then most important issues in building crowd sourcing systems such as recruiting and retaining users, define tasks for the users, combining results provided by users and evaluating their results are presented.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Internet Computing

Keywords

ACM proceedings, LaTeX, text tagging

1. OVERVIEW

Crowd sourcing [3], which is utilizing a crowd of humans to solve a task that is typically easy to solve for humans but hard for machines is becoming a well established problem solving method. The following 4 major tasks have to be addressed by a crowd sourcing application to become successful.

- Recruiting and retaining users
- Define what the users should do
- Combining users results
- Evaluating the results

We take a first step in evaluation and categorization of crowd sourcing systems by classifying them according to the type of collaboration and the task performed by contributors.

1.1 User Collaboration

Explicit collaboration requires users to willingly participate in creation of artifacts or solving of problems, such as writing Wikipedia [9] articles or reviewing and voting products on Amazon.

Implicit collaboration means that users are collaborating by working on other tasks such as playing games or browsing websites.

1.2 Tasks

Further classification according to tasks is done by task performed by contributors of the system.

1.2.1 Explicit Systems

Evaluation Systems let users analyze and comment on items such as books, movies or web pages. The result can be given in textual comments, number scores or tags.

Sharing Systems let users share items such as products, services or textual and structured knowledge. Systems that share products and services [5] are programmableweb.com, YouTube or CPAN. Systems that share textual knowledge are QA web sites (such as Yahoo Answers [1]) and customer support systems (such as Ask Jeeves' AnswerPoint). Systems that share structured knowledge include Swivel, Google Fusion Tables [4] and many science web sites (such as galaxyzoo.org [10]). Sharing systems can be centrally organized such as YouTube or working in a peer-to-peer fashion such as Orchestra.

Building Artifact Systems They allow users to build textual knowledge bases, structured knowledge bases and software. Example of a textual knowledge base is Wikipedia where users contribute texts and web pages as well as edit and merge each others contributions. Structured knowledge bases include DBpedia [8] that is based on Wikipedia info boxes, Freebase.com [2] that builds a structured database and Google Fusion Tables that let users contribute tabular data and collaborate on it by merging tables and commenting on them. Building a structured knowledge base requires selecting a set of data sources, extracting data from them and integrating and merging them. Users can contribute by selecting data sources, integrating structured data, resolving ambiguities and adding inferences. The most well known artifact besides Wikipedia is open source software where users contribute by writing source code, documentation and test of the software.

Task Execution Systems let users perform collaborative tasks such as searching for missing people on images or running election campaigns to mobilize voters. A number of crowd sourcing platforms on the web such as Amazons Mechanical Turk. [7] help distribute pieces of tasks to a crowd and collect the results. Figure 1 shows the different kinds of explicit crowd sourcing systems.

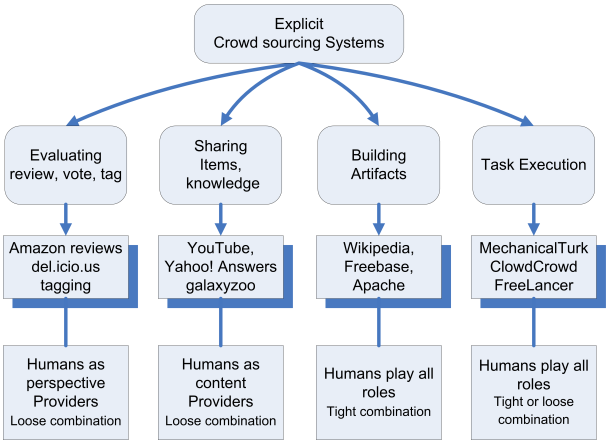

Figure 1: Explicit Systems

1.2.2 *Implicit Systems*

Standalone Systems provide a service so that users utilizing this service collaborate to solve a problem. E.g. the ESP game [11] lets users play a game of guessing common words that describe images, those are then used to label the images. IMDB [6] lets users import movies in a way that it motivates them to rate these movies. reCAPTCHA [12] asks users to solve captchas to prove that they are human and uses the results for handwritten digit recognition.

Piggyback Systems utilize user traces of other systems to perform a task. Such systems have been built on top of search engines such as Google or Bing to predict epidemics or find synonyms. Other examples use traces of purchases to recommend products or use click logs to improve presentation of a web site. Figure 2 shows the different kinds of implicit crowd sourcing systems.

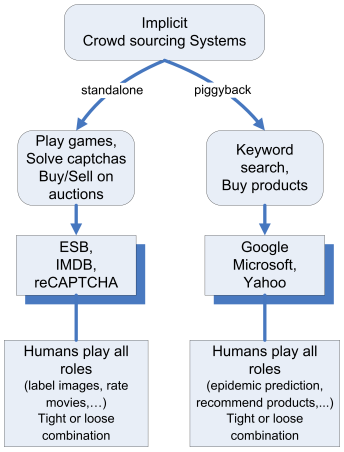

Figure 2: Implicit Systems

2. KEY ISSUES

2.1 Recruiting and Retaining Users

This is the most important crowd sourcing challenge to solve when running such a system. Following solutions can be distinguished.

- *Requires Users* to make contributions. E.g. in an in house system, management may order users to enter their contributions.
- *Pay Users* This is a very simple solution but not effective in economic sense if the services provided by the users are not very valuable. e.g. Amazons Mechanical Turk provides a way to pay users.
- *Ask for Volunteers* This is the most popular solution since it is free and easy to execute but it is hard to predict and achieve that enough contributors are found for an application.
- *Let Users "pay" for the Service* This works for implicit systems where users are only allowed to use the real service behind if they have contributed to the crowd sourcing system before.
- *Piggyback on Existing Systems* This gives a steady stream of users but can only be applied to problems that can be solved by existing user traces.

When the users have been acquired the next step is to encourage and retain them. Popular schemes are *Instant Gratification* to give users immediate feedback on their contributions and show them the positive effects, *Enjoyable Experience* e.g. to allow playing a game for a contribution,*Reputation* to establish, measure and show fame and trust, this can be a very strong means to encourage users,*Set up Competitions* e.g. show top rated users and provide *Ownership Situations* where users feel they own part of the system.

2.2 User Contributions

In many crowd sourcing systems contributions that users can make are limited to simple tasks like reviewing, rating or tagging but in more complex systems users do much more cognitively challenging tasks, e.g providing inference rules or resolving controversial issues in building structured knowledge bases. Often users are classified in groups such as guests, regulars, editors and administrators. Low ranking users provide easy contributions such as editing a paragraph or flagging incorrect ones whereas high ranking users take the hard parts.

Roles humans play in a crowd sourcing solution are

- *Slaves* humans solve a problem in divide and conquer fashion to utilize resources.
- *Perspective Providers* combine different perspectives such as in reviewing books or movies.
- *Content Providers* provide user generated content such as movies in YouTube, pictures in Flickr or articles in Wikipedia.

- *Component Providers* humas work as components in a complex system such as a social network.

Often in real applications users play several roles together (e.g. perspective providers and content providers in Wikipedia).

In addition it is important to quantify the impact user contributions have on the overall artifact. Flagging an incorrect piece of data has far fewer impact than contributing a wrong inference rule. Only high ranking users should be allowed to make such contributions. Not all contributions in a crowd sourcing system will come from humans, often a split of tasks between them and machines is performed. Humans should concentrate on tasks where they outperform machines, e.g. determining whether two textual or image descriptions are equal to each other.

Finally the user interface should be optimized for the tasks performed by users. Natural language input (e.g. in openmind.com)is easy for users but hard to translate to the crowd sourcing application, formal language is just the other way around.

2.3 Combining User Results

Many crowd sourcing systems do not combine contributions or combine them only in a simple way using numeric formulas. Other systems, e.g. those that build structured knowledge bases or software employ more sophisticated concepts. A key issue is what to do if user contributions differ, e.g. one user is asserting "A" and the other "not A". Automatic solutions employ voting mechanism where contributions are weighted with the trustworthiness of each user. Manual systems let users resolve differences by themselves, unresolved issues are escalated to higher ranking users. They are still the preferred way to solve difficult issues. Especially, if also machines make contributions, resolving issues can become very tricky since machines cannot participate in human discussions to resolve it.

2.4 Evaluating User Results

Crowd sourcing systems have to manage malicious users. Typical techniques are to detect, block and deter them, e.g. only trustworthiness users are allowed to make high impact contributions. Malicious users and contributions can be detected by manual techniques, e.g by users flagging bad contributions. Automatic systems ask e.g. questions where the answer is already known and then uses the users answer to compute a reliability measure. Other automatic systems base a users trustworthiness on the fame and reputation of the user. Malicious users may also be deterred by "public shaming" where such a user is flagged in view of the other users. Another track is to detect users can provide exceptionally good results and get them rewarded and promoted to higher ranks.

If bad content has been inserted into a crowd sourcing system there must be ways to undo it. E.g. in Wikipedia changes are local and logged and can be undone if a malicious contribution is detected. In other cases, such as knowledge bases with inference rules it is much harder to undo changes that had already influenced other contributions without reverting completely back.

3. CONCLUSIONS

Crowd sourcing system can be applied to variety of real world problems and it can be expected that this field will grow strongly in the future. It can be expected that generic platforms will emerge that allow to build applications much easier and faster. In addtion crowd sourcing will be applied to more formal and structured problems such as building structured databases and services. Platforms will also make not much experienced users to make more valuable contributions such as building Mashups and complex structured data items.

4. REFERENCES

[1] L. A. Adamic, J. Zhang, E. Bakshy, and M. S. Ackerman. Knowledge sharing and yahoo answers: everyone knows something. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 665–674, New York, NY, USA, 2008. ACM.

[2] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, New York, NY, USA, 2008. ACM.

[3] D. C. Brabham. *Crowdsourcing.* The MIT Press essential knowledge series. MIT Press, 2013.

[4] H. Gonzalez, A. Halevy, and C. Jensen. Google Fusion Tables: Data Management, Integration and Collaboration in the Cloud. In *SoCC*, pages 175–180, 2010.

[5] D. Grewing and D. Nguyen Nguyen. Java sozial - social apis, opensocial und die java welt. *Java Magazin*, (11):48 – 54, 2010.

[6] Y. Hu, Z. Wang, W. Wu, J. Guo, and M. Zhang. Recommendation for movies and stars using yago and imdb. In W.-S. Han, D. Srivastava, G. Yu, H. Yu, and Z. H. Huang, editors, *APWeb*, pages 123–129. IEEE Computer Society, 2010.

[7] P. G. Ipeirotis. Analyzing the amazon mechanical turk marketplace. *ACM Crossroads*, 17(2):16–21, 2010.

[8] J. Lehmann, J. SchÃijppel, and S. Auer. Discovering unknown connections - the DBpedia relationship finder. In *Proceedings of the 1st Conference on Social Semantic Web (CSSW 2007)*, volume 113 of *LNI*, pages 99–110. GI, 2007.

[9] G. Schuler. *Wikipedia inside.* 2007.

[10] University of Oxford. Galaxy zoo opens, July 2007.

[11] L. von Ahn and L. Dabbish. Esp: Labeling images with a computer game. In *AAAI Spring Symposium: Knowledge Collection from Volunteer Contributors*, pages 91–98. AAAI, 2005.

[12] L. Von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum. reCAPTCHA: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468, 2008.