# A Framework for Crowdsourced Multimedia Processing and Querying

Alessandro Bozzon Ilio Catallo Eleonora Ciceri
Piero Fraternali Davide Martinenghi Marco Tagliasacchi
Dipartimento di Elettronica e Informazione
Via Ponzio 34/5
Milan, Italy
first.last@polimi.it

## ABSTRACT

This paper introduces a conceptual and architectural framework for addressing the design, execution and verification of tasks by a crowd of performers. The proposed framework is substantiated by an ongoing application to a problem of trademark logo detection in video collections. Preliminary results show that the contribution of crowds can improve the recall of state-of-the-art traditional algorithms, with no loss in terms of precision. However, task-to-executor matching, as expected, has an important influence on the task performance.

## Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

## General Terms

Design, Experimentation, Human Factors, Measurement, Performance

## Keywords

Human Computation, Crowdsourcing, Multimedia

## 1. INTRODUCTION

Human computation is an approach to problem solving that integrates the computation power of machines with the perceptual, rational or social contribution of humans [6]. Within human computation, crowdsearching can be defined as the application of the principles and techniques of human computation to information retrieval, so as to promote the individual and social participation to search-based applications and improve the performance of information retrieval algorithms with the calibrated contribution of humans. Traditionally, crowdsearching methods in information retrieval have been exploited to address problems where humans outperform machines, most notably common sense knowledge elicitation and content tagging for multimedia search [3]. In the latter field, crowdsourced multimedia content processing exploits the fact that humans have superior capacity for understanding the content of audiovisual materials, and thus *replaces* the output of automatic content classification with human annotations, and feature extraction with human-made tags.

In this paper, we adopt a different approach: rather than replacing feature extraction algorithms, we aim at *improving their performance* with well-selected tasks assigned to human executors, thus realizing a more integrated collaboration between human judgement and algorithms.

The contribution of the paper is the illustration of the design, implementation, and preliminary evaluation of a crowdsearching application for trademark logo detection in video collections, in which the help of the crowd is sought for selecting the most appropriate images associated with a brand name, so as to improve the precision and recall of a classical image retrieval approach based on local features (e.g. SIFT) [5].

The paper is organized as follows: prior to introducing the logo detection application, in Section 2 we present a high-level conceptual framework that helps in characterizing the problems that need to be faced in crowdsearching application development. Next, Section 3 explains the design and implementation of the logo detection application, while Section 4 reports on an experimental evaluation that compares three scenarios of image to video matching: one completely automated, one with expert support to task execution, and one with the help of generic facebook users. Finally, Section 5 concludes with an illustration of the ongoing efforts for implementing and evaluating the utility of a crowdsearch platform capable of assisting the development of a broad variety of crowdsearching solutions.

## 2. A FRAMEWORK FOR HUMAN COMPUTATION

In any human computation approach to problem solving, and hence also in crowdsearching, a problem is mapped into a set of tasks, which are then assigned to both human and machine executors in a way that optimizes some quality criterion on the problem-solving process, like, e.g., the quality of the found solution or the time or money spent to find it.

Figure 1 conceptualizes the steps that lead from the formulation of a complex problem solving goal to its execution


*CrowdSearch* 2012 Lyon, FR

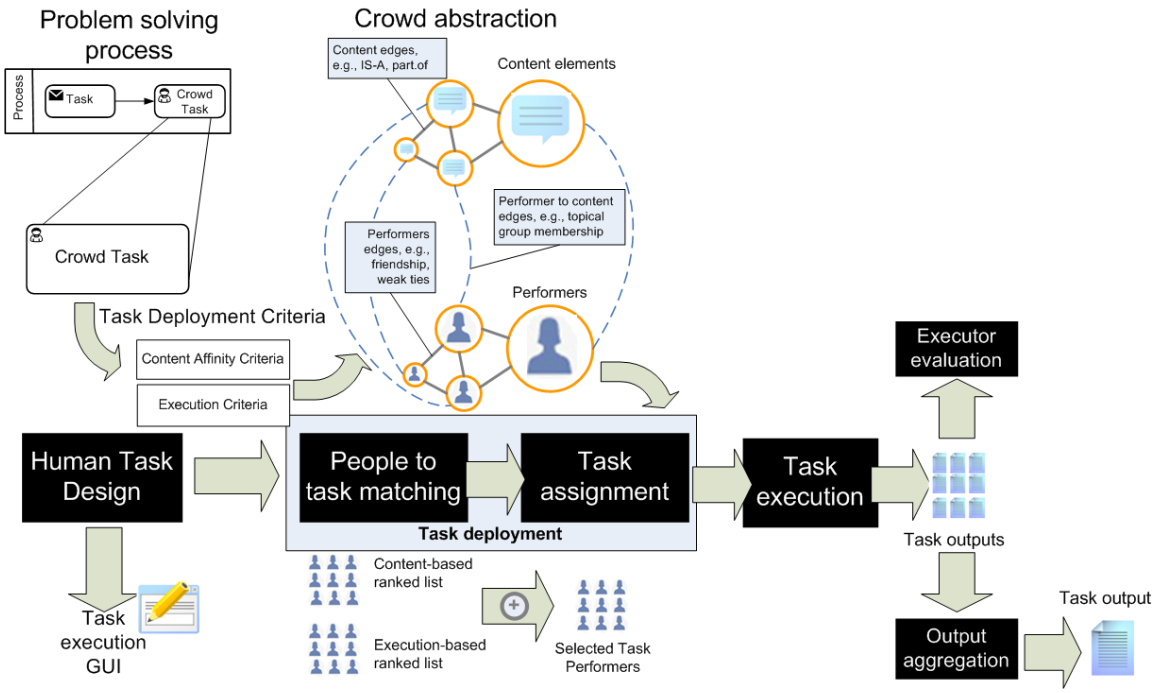


**Figure 1: The CrowdSearch Framewok**

and verification with the help of a crowd of executors.

The entry point is the specification of a problem solving process, defined as a workflow of tasks that leads to a desired goal. Such a notion is purposely broad and embraces both general-purpose cooperative and distributed processes, as found, e.g., in Business Process Management (BPM), and more focused problem instances, like crowd-supported multimedia feature extraction. The common trait is that multiple tasks must be executed respecting precedence constraints and input-output dependencies, and that some of the tasks are executed by machines and some by an open-ended community of workers (the latter are named Crowd Tasks in Figure 1). Unlike in classic BPM, the community of workers is not known a priori, and the task assignment rules are dynamic and based on the tasks specification and on the characteristics of the potential members of the crowd.

A Crowd Task is the subject of Human Task Design, a step that has the objective of defining the modalities for crowdsourced task execution. Human Task Design produces the actual design of the Task Execution GUI, and the specification of *Task Deployment Criteria*. These criteria can be logically subdivided into two subject areas: Content Affinity Criteria (what topic the task is about) and *Execution Criteria* (how the task should be executed). The Content Affinity Criteria can be regarded as a query on a representation of the users' capacities: in the simplest case, this can be just a descriptor denoting a desired topical affinity of the user (e.g., *image geo-positioning*): in more complex cases it can be a semi-structured data query (e.g., a set of attribute-value pairs, like *action=translation from=English to=Italian*), or a query in a logical language.

The Execution Criteria could specify constraints or desired characteristics of task execution, including: a time budget for completing the work, a monetary budget for incentivizing or paying workers, bounds on the number of executors or on the number of outputs (e.g., a level of redundancy for output verification) and desired demographic properties (e.g., workers' distribution with respect to geographical position or skill level).

Symmetrically to the problem solving process, also the crowd of potential performers and their capacities have to be abstracted. A natural *Crowd Abstraction* is a bi-partite graph (as shown in Figure 1), where nodes denote either *performers* or *content elements*, and edges may connect performers (to denote, e.g., friendship), content elements (to denote semantic relationships) and performers to content elements (to denote, e.g., interest). The bi-partite graph representation can be refined by attaching semantics to both nodes and edges: users can be associated with profile data; content elements can be summarized or classified in topics; performer edges can express explicit friendship or weak ties due to interactions between users; content element edges can express ontological knowledge, like classification, part-of, etc.; user to content edges can represent a specific capability (e.g., ability to review, produce, or judge about content elements).

The subsequent *Task Deployment* step comprises the selection of the candidate performers, by *People to Task Matching*, and then the *Task Assignment* to a set of actual performers. The People to Task Matching can be abstracted as a query matching problem, in which the Task Deployment Criteria are used to extract from the crowd abstraction graph a ranked list of potential candidate workers ranked according to their expected suitability as task executors. In the most general case, the measure of suitability is composed of a part that embodies the Content Affinity Criteria of the candidates to the task (e.g., user-to-task topical similarity) and a part that measures the appropriateness of a

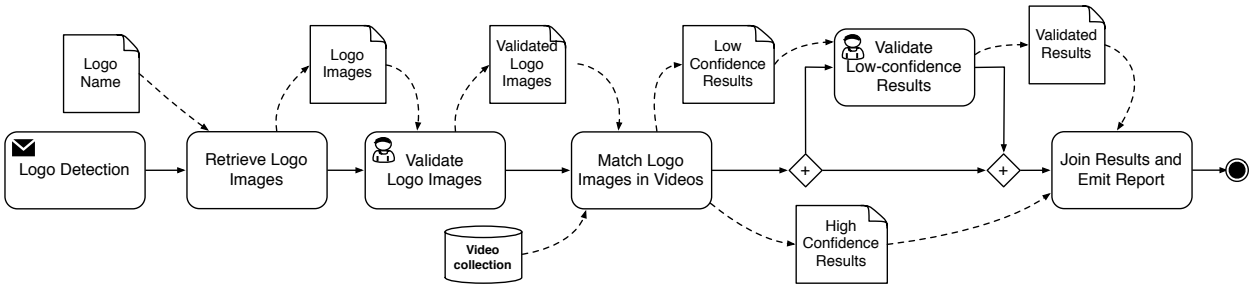


**Figure 2: The Process of the Logo Detection Application**

candidate, or of a set of candidates, to satisfy the Task Execution Criteria. Evaluating the People to Task Matching under the Execution Criteria can be a complex achievement that requires addressing different aspects, such as the match between task difficulty and skill level, the role and influence of users in the network (which determines their ability to spread the task), and so on. Then, the topical and execution suitability measures should be combined to obtain a globally good set of candidates. This can be regarded as the aggregation of a content-based and of an execution-based ranked list of potential candidates, of which the top-k ones are selected for the actual task assignment.

As an example, task-to-performer assignment can be formulated as a matching problem in a vector space, by representing both the Task Deployment Criteria and the candidate Performers as feature vectors in a space of appropriate dimensionality and then computing the match score using vector similarity measures. The problem could be further modularized by decomposing the task description vector into two components: one denoting the Content Affinity Criteria and one denoting the Execution Criteria. The result sets for these two queries could then be merged with a rank aggregation approach.

The *Task Execution* step represents the actual execution of the task by the selected performers, which results in multiple outputs; these are then aggregated to form the final outcome of the Crowd Task. As usual in crowdsourcing, redundancy can be exploited to cope with the uncertain quality of the worker's performance. In this case, multiple outputs for the same task must be merged to obtain a final task output with a high level of confidence. As a result of evaluating the task output, feedback can be generated on the skill level of performers (by the *Executor Evaluation* phase).

# 3. THE LOGO DETECTION APPLICATION

In Section 2 we presented a framework for crowdsourced multimedia processing and querying, that requires the crowd to execute tasks during a Problem Solving Process. In this Section we illustrate an example of Problem Solving Process, for which we have designed, deployed and evaluated one Crowd Task: the Logo Detection application.

## 3.1 Specifications

As a proof of concept, we have designed a problem solving process for trademark logo detection in video collections. The goal is to receive from a user a query consisting of a brand name and to produce a report that identifies all the occurrences of logos of that brand in a given set of video files. The logo detection problem is a well-known challenge in image similarity search, where local features, e.g., SIFT, are normally employed to detect the occurrences of generic object based on their scale invariant properties [5]. However, image similarity based on SIFT is largely affected by the quality of the input set of images used to perform the matches, which makes it an interesting case for introducing the contribution of humans.

Therefore, we have designed a crowdsearching application that consists of a sequence of both automated tasks and crowd tasks, illustrated in Figure 2, with the goal of increasing precision and recall with respect to fully automated solutions. Specifically, the crowd contribution is exploited on two levels: for retrieving good images that well represent the brand name to be searched; and for validating matches of the logos in the video collection for which the content-based image retrieval based on SIFT descriptors has reported a low matching score.

The process receives as input a textual keyword indicating the name of the brand. The first task (*Retrieve Logo Images*) performs text-based image retrieval to associate a set of representative logo images to the brand name string; this task can be executed by an automated component (in our implementation, we have used the Google Images APIs[1]). However, as we will see, the output of Google is far from perfect, as the search engine returns images based on the textual content of the page that contains them, which determines the presence of many false positive and low quality images in the automatically constructed result set.

The next task (*Validate Logo Images*) is a crowd task: it employs human computing in order to assist the validation of the images retrieved by Google Images, so as to enhance the performance of the content-based image retrieval based.

Then, an automated task (*Match Images in Videos*) looks for occurrences of the different versions of the logos in a collection of video clips, using a content-based image retrieval component (i.e., the OpenCV library [2] implementing the SIFT algorithm [5]). The output is a list of triples: <*videoID*, *frameID*, *matchingScore*>, where *matchingScore* is a number between 0 and 1 that expresses how well the searched logo has been matched within the frame (identified by *frameID*) in the video identified by *videoID*.

The process continues with a second crowd task (*Validate Low Confidence Results*), which dispatches to the crowd those matches that have a matching score lower than a given threshold. Finally, the result report is constructed (*Join Results and Emit Report* Task) by adding the high score

[1] `http://images.google.com/`

matches found by the algorithms and the low score matches manually validated by the users.

## 3.2 An Architecture for Crowd Task Execution

In order to enable the deployment of applications that comprise crowd tasks, such as the one described in Section 3.1, we are building the technical architecture for performer and task management illustrated in Figure 3. At present, we have implemented and deployed the crowd task *Validate Logo Images*.

The architecture of Figure 3 supports the creation of a task and of the associated GUI, its assignment to a pool of performers through a crowd task execution platform, and the collection of the task output. We envision three major task execution modalities: structured crowdsourcing platforms (e.g., Microtask.com and Mechanical Turk), open social networks (e.g., Facebook and G+) and email invitations to perform the task using a custom Web application. At present, the implementation uses Facebook as a crowd task execution platform.

The *Task GUI and Criteria Generation* is an online application that generates a task GUI and some simple deployment criteria from a *Task Template*, i.e., an abstract description of a piece of work, characterized by the following dimensions: task type (open/closed question, custom Web application), task description, deployment criteria (none, by location, by skill, by similar work history), and task alternates (i.e., variants of the same task specification that can be associated with specific deployment criteria, e.g., with different skill levels). In the logo detection application, we have designed a task template with two variants. In the base variant for novice users, the task GUI presents a brand name and a set of images taken from Google Images (see Figure 5), with checkboxes for choosing the images that best match the brand name. In the second variant, aimed at people that have done at least one instance of the basic variant of the task, the challenge is to input the URLs of new (and possibly better) images that match the brand name (see Figure 6).

The *Task Deployment* function consists of:

- *Task to People Matching* lets the task owner connect to a number of community platforms and collect candidate performers in *worker pools*. Worker pools can be also edited manually, like a contact list; to ease selection, candidates in a pool can be ranked w.r.t. the deployment criteria of a task, based on the available candidate profile data and work history information. The present implementation of this functionality is a native Facebook application that enables the task owner to assign candidate performers to the worker pool by picking them from his list of friends.

- *Task assignment* allows the task owner to create an instance of a task and dispatch it, with the associated GUI, to the Task Execution platform. The task submission occurs differently based on the target platform: it may be a post on the user's space in a social network, a personal invitation email message, or the task publication in a crowdsourcing platform. The present implementation is the same Facebook application used for Task to People Matching that supports the dispatch of a task instance (i.e., a brand name for which images must be validated) in the form of a post on the performer's wall (as shown in Figure 4).

The *Crowd Task Execution* step is implemented as a native Facebook application that the performer has to install in order to perform the task, and, if he wishes, to re-dispatch it to friends.

The *Task output collection and aggregation* collects the output of the task instances that have been dispatched by the task owner. producing an unified view on the retrieved values. In the logo detection application, for each brand name it returns 1) the new images suggested by the performers, and 2) for each image from Google Images, the number of accorded preferences.

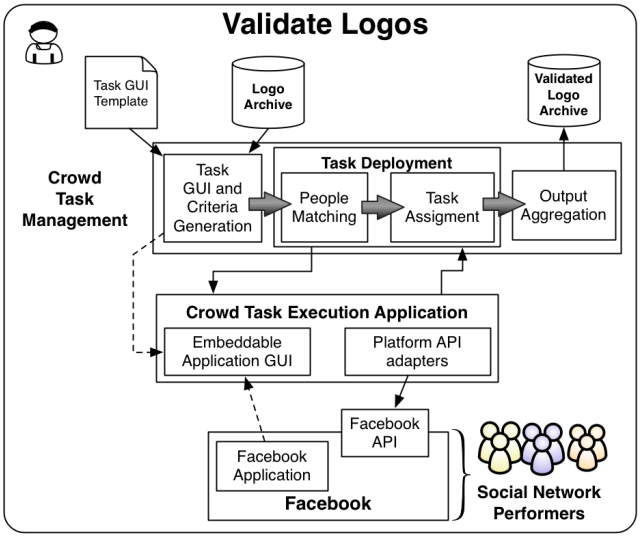


**Figure 3: Architecture for Crowd Task management.**

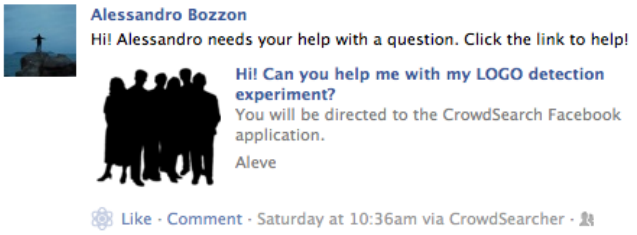


**Figure 4: UI of the Facebook task invitation message.**

## 3.3 Status of the implementation

The logo application is currently based on the CrowdSearcher task distribution and execution platform [1]. The system has been configured to 1) create task templates by connecting with the logo detection application to retrieve the set of logos being evaluated, 2) send queries to the Facebook platform, 3) select the set of workers to be involved in the task, and 4) gather the results. Being deployed on a social network platform, CrowdSearcher acts in the context provided by a given Facebook user, who is instrumental to the crowd-sourcing process, being responsible of initiating


**Figure 5: UI of the task execution interface: logo selection.**


**Figure 6: UI of the task execution interface: logo insertion.**

the tasks which are spawn to the crowd, and by offering friends and colleagues as workers.

The Facebook application embeds a platform-specific client which communicates with the CrowdSearcher server. The client serves a twofold purpose. On one hand, it allows workers (Facebook users) to perform deployed tasks, as depicted in Figure 5 and 6. On the other hand, the application exploits the native Facebook Graph API to enable a user-defined worker selection, where new workers are explicitly invited by their friends; Figure 4 depicts an example of task invitation performed on the Facebook wall of a targeted user. The choice of allowing a manual worker selection is supported by the findings in [1], where it is shown how a higher participation can be achieved when workers are manually picked by the user.

Tasks are assumed to have a timeout for completion, specified in the logo identification application, that defines how long the system should wait for human execution. When the timeout triggers, the system automatically aggregates the task results – respectively, the number of preferences for each logo image, and the URL of the newly provided logo images – feeding the validated logos archive, which is next used for the matching of logo images in the video collection.

## 4. EXPERIMENTAL EVALUATION

In this section we present the preliminary experiments that we conducted on a public video collection [4] containing 120 grocery product logos, to compare automated and crowd-supported logo detection. Experiments involving humans have been performed on three medium-sized logos (Aleve, Chunky and Claritin). We tested the performance of the process in Figure 2 using the standard measures of *precision* and *recall* on the output of *Match Logo Images in Video* task.

The CrowdSearcher system[2] has been adopted to support the *Validate Logo Images* crowd task, by allowing users to 1) *select* existing logos (Figure 5) to improve the precision of the application by providing the matching component with correct logo instances, and to 2) *add* new logos (Figure 6), with the purpose of increasing the overall recall by adding novel logo examples.

Around 40 people were involved as workers for the selection or provision of image logos, mostly from the students in our classes, or student's friends who volunteered to be part of the experiment. Some 50 task instances were generated in a time-span of three days, equally distributed on the set of considered logos, resulting in 70 collected answers, 58% of which related to logo images selection tasks.

We tested performance under three experimental settings. (1) No human intervention in the logo validation task: here, the top-4 Google Images result set is used as a baseline for the logo search in the video collection; the result set may contain some irrelevant images, since they did not undergo validation. (2) Logo validation performed by a crowd of domain experts (simulation): the top-32 Google Images results are filtered by experts, thereby deleting the non-relevant logos and choosing three images among the relevant ones. (3) Inclusion of the actual crowd knowledge: filtering and expansion of the set of matched logos is done via the CrowdSearcher application.

The results are shown in Table 1. For each logo, precision and recall are evaluated for the three versions of the application.

Expert evaluation clearly increases the application performance in both precision and recall, with respect to a fully-automated solution. This is due to the fact that the validation process performed on the set of logo instances eliminates irrelevant logos from the query set, and consequently reduces the number of false positives in the result set. On the other hand, the validation conducted by the crowd showed generally a slight increase in both precision and recall. However, the performance increase is not evenly distributed over all logo brands: we believe that this is due to a different user behavior in the choice of the relevant im-

[2]Available at https://apps.facebook.com/crowd_search/

| Brand name | Test | Precision | Recall |
|---|---|---|---|
| Aleve | No Crowd | 0.27 | 0.27 |
| | Experts | 0.42 | 0.54 |
| | Crowd | 0.33 | 0.41 |
| Chunky | No Crowd | 0.65 | 0.19 |
| | Experts | 0.70 | 0.58 |
| | Crowd | 0.40 | 0.21 |
| Claritin | No Crowd | 0.31 | 0.09 |
| | Experts | 0.57 | 0.72 |
| | Crowd | 0.36 | 0.73 |

**Table 1: *Precision* and *Recall* of the logo detection application in the three considered experimental settings.**

age set. In particular, when validating the *Chunky* brand logos, the crowd chose within the top-2 image an irrelevant logo, as shown in Figure 7. Consequently, the performance has been affected, with a heavy decrease both in terms of precision and recall w.r.t. the expert evaluation. This result brings to a consideration about the context of human enacted executions: the chances to get good responses depend on the appropriateness of the users' community w.r.t the task at hand. Both the geographical location and the expertise of the involved users can heavily influence the outcome of human enacted activities, thus calling for a fine-grained task to people matching phase.


**Figure 7: An example of a) correct and b) wrong logo for the Chunky brand**

## 5. DISCUSSION

We have presented a framework and an architecture for handling task design, assignment and execution in crowd-empowered settings. We have then described a trademark logo detection application that can be easily accommodated in the presented framework and whose execution can largely benefit from the presence of a crowd of users. Our initial experiments have shown that human-enriched tasks, such as logo validation and insertion, contribute to a non-negligible improvement of both recall and precision in the obtained result set. Yet, such an improvement is unevenly distributed over the different queries we tried, mostly because some users did not have an adequate background to answer the questions that were sent to them. This suggests that users should be more carefully selected during task assignment. Future directions of research therefore include studying how to associate the most suitable request with the most appropriate user, so as to implement a ranking function on worker pools whereby the task owner is aided in the dispatch of the task to the top-k best candidates.

Along the same lines, we are also studying a crowdsearch scenario in which engineering the most suitable task/request plays a crucial role. Here, the end user wants to reconstruct the correct temporal sequence of user-generated videos regarding particular events (e.g., breaking news). At peak moments, there may be a proliferation of such videos by means of reposting and re-editing of their original content, which makes the problem non-trivial. Indeed, the actual creation date of a video clip (as indicated by tags in the file) may be uncertain and thus unreliable, thereby producing ambiguities in the temporal ordering of the clips, much in the same way in which, in top-k queries, uncertain scores determine multiple possible rankings [7]. While finding temporal dependencies between two clips may in some cases be done automatically by detecting near duplicates of a video, fully reconstructing the temporal sequence will require human intervention. Humans will assist the process by i) resolving conflicts between the creation dates available in the tags and the temporal dependencies inferred by near duplicate detection, and ii) refining the time interval associated with a video clip's creation date. In our research agenda, emphasis will be placed on the following two aspects. (1) Tasks will be engineered in such a way that their resolution maximizes the expected decrease of the amount of uncertainty associated with the temporal ordering of the clips. (2) We shall give priority to reducing uncertainty of videos close to a certain date of interest (typically, the date of the event at hand), thereby focusing on the ordering of the "top-k" such videos.

**Acknowledgments** This work is partially supported by the FP7 Cubrik Integrating Project[3] and by the Capacities Research for SMEs project BPM4People of the Research Executive Agency of the European Commission [4].

## 6. REFERENCES

[1] A. Bozzon, M. Brambilla, and S. Ceri. Answering search queries with crowdsearcher. In *To appear in Proceedings of the 21st International Conference on World Wide Web, WWW 2012, Lyon, France, April 16-20, 2012*, 2012.

[2] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[3] X. Hu, M. Stalnacke, T. B. Minde, R. Carlsson, and S. Larsson. A mobile game to collect and improve position of images. In *Proc. Third International Conference on Next Generation Mobile Applications, Services and Technologies*, 2009.

[4] U. C. V. Laboratory. Grozi-120 database.

[5] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[6] A. J. Quinn and B. B. Bederson. Human computation: a survey and taxonomy of a growing field. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, CHI '11, pages 1403–1412, 2011.

[7] M. A. Soliman, I. F. Ilyas, D. Martinenghi, and M. Tagliasacchi. Ranking with uncertain scoring functions: semantics and sensitivity measures. In *SIGMOD Conference*, pages 805–816, 2011.

[3] http://www.cubrikproject.eu/
[4] http://www.bpm4people.org/