Human Computation and Crowdsourcing: Works in Progress and Demonstration Abstracts
AAAI Technical Report CR-13-01

Automating Crowdsourcing Tasks in an Industrial Environment

Vasilis Kandylas, Omar Alonso, Shiroy Choksey, Kedar Rudre, Prashant Jaiswal
{vakandyl, omalonso, schoksey, kedaru, prjaiswa}@microsoft.com
Microsoft

Abstract

Crowdsourcing based applications are starting to gain traction in industrial environments. Crowdsourcing research has proven that it is possible to get good quality labels at the fraction of the cost and time. However, implementing such applications at large scale requires new infrastructure. In this demo we present a system that allows the automation of crowdsourcing tasks for information retrieval experiments.

Introduction

In the context of information retrieval systems, crowdsourcing has become increasingly prominent, in particular for evaluating the quality of search results as well as for gathering labeled data for training machine learning algorithms. Getting relevance assessments or labels typically involves employing humans to perform the annotations, which directly translates to training and working costs. Crowdsourcing platforms like Amazon Mechanical Turk (MTurk) have made labeling cheaper and faster but there are hidden costs for implementing tasks, managing quality control and producing high quality data. We are interested in simplifying and automating the process of setting up tasks that involve human computation. We implemented a number of features that are common to many tasks and allow developers to concentrate on the experiment that they are trying to conduct instead of operational details. In this paper we describe a production system by presenting two applications: evaluation and classification.

Description

We developed a tool, BingDAT, that acts as an execution management engine and which can interact with other systems, including Cosmos/Scope (Chaiken et al. 2008), a parallel processing system, and UHRS, an internal crowdsourcing platform (Patel 2012) comparable to MTurk. TurkServer (Mao et al. 2012) provides a similar approach for MTurk. In our solution UHRS is a data source so it is possible to integrate with MTurk as well.

BingDAT is a powerful data pipeline platform that allows developers to easily set up complex data pipelines. Once a pipeline is configured, there is no need to do anything other than review the results. BingDAT (figure 1) incorporates many other functions, such as data manipulation, analysis, historical tracking, reporting, alerting and visualization, but in this demo paper we focus on its use in creating workflows which contain automated and human elements.

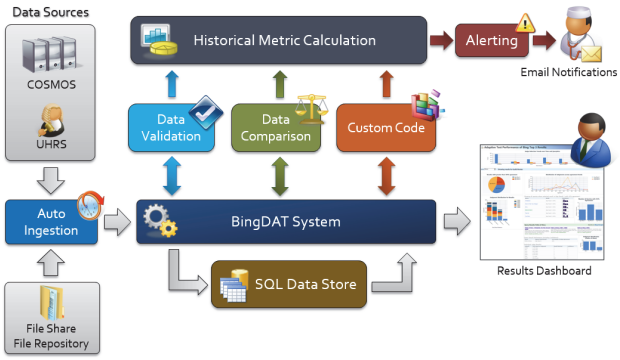

Figure 1: Block diagram of BingDAT.

The workflows are defined by sequencing import, computation and export jobs. A typical workflow consists of one or more input jobs, which receive the data from some sources, computation jobs which process them, and export jobs which output the results. All the jobs and entire workflows can be chained together, so for example BingDAT can import logged search data from Cosmos, process them, send them to UHRS for evaluation, receive the output of the human judges, and display the results. Individual jobs are chained together as building blocks, so they can be easily re-used in other tasks with minimal changes.

Applications

Measurement and Quality Evaluation

A common application is evaluation, where human judges are used to rate items, such as search results. Another is comparison, where human workers are selecting between different items, for example the best of two layouts or the preferred of several displayed results.

BingDAT is well suited for automating repetitive tasks that runs continuously. One such task is monitoring the relevance of the search results (figure 2). In that case, Bing-

DAT will get data (step 1), either by scraping the production search engine or from data logged in Cosmos. Then it will process and transform the data (step 2) and export them to UHRS for human evaluation (step 3). Once the human task is finished, BingDAT will import the results from UHRS (step 4) and process them. At this stage BingDAT will perform typical crowdsourcing computations (for example. remove workers' answers that were detected as spam, or deal with disagreements) and also calculate metrics of the result relevance, the judge performance and possibly send alerts if some results exceed certain thresholds (step 5). Finally, the results are visualized in a dashboard page (step 6). The whole process can be configured to be repeated automatically at certain time intervals (e.g. daily, weekly etc.). The users of BingDAT do not need to do anything more than check the dashboard page or just check for any alerts they may receive.

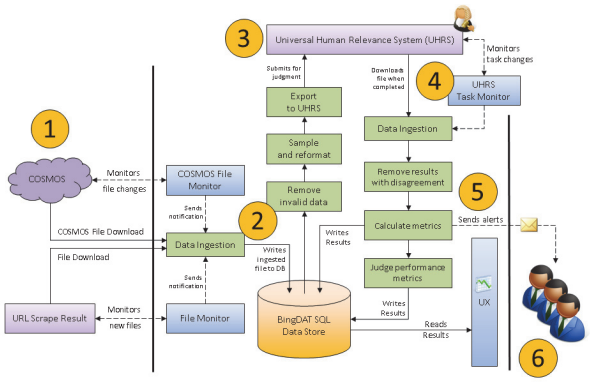

Figure 2: Relevance evaluation pipeline implemented with BingDAT.

Other measurement or comparison tasks operate in a similar manner. For example, to perform side-by-side comparisons, BingDAT is configured to get the required data from the appropriate source and submit them to UHRS, where the human workers would choose their preferred side (left or right) and BingDAT then computes and displays the results.

ML Model Verification

Another application is to use human workers to verify the output of a machine-learned model. An example is a classifier is trained to identify a certain kind of spam web sites, which are then blocked from appearing in the results. Relying only on the classifier output is risky, because any false positives would be blocked, causing good web sites to disappear from the results. We therefore add a second verification step that relies on humans. The specific kind of spam sites is rare, so the number of positives that the classifier returns can be easily verified by humans. BingDAT is configured to get the classifier positive output, send it to UHRS for human verification, process the results and remove any false positives that the judges identified, and then submit the final, verified positives for blacklisting.

Judge Performance Monitoring

We have created jobs in BingDAT to monitor human worker performance. These jobs are used in any task where humans are involved and give developers an overview of the quantity and the quality of their work. Figure 3 shows an example scorecard for a judge for a side-by-side comparison task. The top left area shows the quality of the judge's work in comparison with the average: how often they agree with the majority, their work progress and their mean square error. The top right chart shows the distribution of the judge's answers (in blue) next to the average distribution of answers (in yellow). This specific judge always gives the same answer in all Human Intelligence Tasks (the single blue bar) whereas the other judges' answers are more evenly distributed. This shows that this judge is either a bot or a spammer and can be reliably removed. The bottom part shows statistics about the judge's daily progress, and the time they take to submit an answer to each HIT.

Figure 3: Sample worker scorecard.

Other metrics available in the dashboard (not shown due to limited space) include confusion matrices showing individual judge answers vs. average judge answers, judge bias to specific answers and judge agreement metrics such as Fleiss' kappa and Krippendorff's alpha.

Discussion

In this paper we described an end-to-end solution for deploying crowdsourcing tasks at large scale. In contrast to MTurk or CrowdFlower that offer a large pool of workers with some level of work quality, our approach makes emphasis on data integration with other systems, built-in quality controls and dashboard features.

References

Chaiken, R.; Jenkins, B.; Larson, P.-A.; Ramsey, B.; Shakib, D.; Weaver, S.; and Zhou, J. 2008. Scope: easy and efficient parallel processing of massive data sets. *Proc. VLDB Endow.* 1(2):1265–1276.

Mao, A.; Chen, Y.; Gajos, K.; Parkes, D.; Procaccia, A.; and Zhang, H. 2012. TurkServer: Enabling Synchronous and Longitudinal Online Experiments. *Proc. HCOMP.*

Patel, R. 2012. Crowdsourcing at Bing. http://research.microsoft.com/en-us/um/redmond/events/fs2012/presentations/rajesh_patel.pdf.