Combining Crowd and Machines for Multi-predicate Item Screening

EVGENY KRIVOSHEEV, University of Trento, Italy
FABIO CASATI, University of Trento, Italy and Tomsk Polytechnic University, Russia
MARCOS BAEZ, University of Trento, Italy and Tomsk Polytechnic University, Russia
BOUALEM BENATALLAH, UNSW, Sydney, Australia

This paper discusses how crowd and machine classifiers can be efficiently combined to screen items that satisfy a set of predicates. We show that this is a recurring problem in many domains, present machine-human (hybrid) algorithms that screen items efficiently and estimate the gain over human-only or machine-only screening in terms of performance and cost. We further show how, given a new classification problem and a set of classifiers of unknown accuracy for the problem at hand, we can identify how to manage the cost-accuracy trade off by progressively determining if we should spend budget to obtain test data (to assess the accuracy of the given classifiers), or to train an ensemble of classifiers, or whether we should leverage the existing machine classifiers with the crowd, and in this case how to efficiently combine them based on their estimated characteristics to obtain the classification. We demonstrate that the techniques we propose obtain significant cost/accuracy improvements with respect to the leading classification algorithms.

ACM Reference Format:
Evgeny Krivosheev, Fabio Casati, Marcos Baez, and Boualem Benatallah. 2018. Combining Crowd and Machines for Multi-predicate Item Screening. *Proc. ACM Hum.-Comput. Interact.* 2, CSCW, Article 97 (November 2018), 18 pages. https://doi.org/10.1145/3274366

1 BACKGROUND AND MOTIVATION

A frequently occurring classification problem consists in identifying items that pass a set of *screening tests* (filters). This is not only common in medical diagnosis but in many other fields as well, from database querying - where we filter tuples based on predicates [32], to hotel search - where we filter places based on features of interest [23], to systematic literature reviews (SLR) - where we screen candidate papers based on a set of exclusion criteria to assess whether they are in scope for the review [42]. The goal of this paper is to understand how, given a set of trained classifiers whose accuracy may or may not be known for the problem at hand (for a specific query predicate, hotel feature, or paper topic), we can combine machine learning (ML) and human (H) classifiers to create a hybrid classifier that screens items efficiently in terms of cost of querying the crowd, while ensuring an accuracy that is acceptable for the given problem. We focus specifically on the common scenario of *finite pool* problems, where the set of items to screen is limited and where therefore it may not be cost-effective to collect sufficient data to train accurate classifiers for each specific case. To make the paper easier to read and the problem concrete, we will often take the

Authors' addresses: Evgeny Krivosheev, University of Trento, Italy, evgeny.krivosheevlast@unitn.it; Fabio Casati, University of Trento, Italy and Tomsk Polytechnic University, Russia, fabio.casati@unitn.it; Marcos Baez, University of Trento, Italy and Tomsk Polytechnic University, Russia, marcos.baez@unitn.it; Boualem Benatallah, UNSW, Sydney, Australia, boualem@cse.unsw.edu.au.

2573-0142/2018/11-ART97 $15.00
https://doi.org/10.1145/3274366

example of SLRs mentioned above, which is rather challenging in that each SLR is different and each filtering predicate (called *exclusion criterion* in that context) could be unique to each SLR (e.g., "exclude papers that do not study adults 85+ years old").

The area of crowd-only and of hybrid (ML+H) classification has received a lot of attention in the literature. Research in crowdsourcing has identified how to best classify items given crowd votes [9, 25, 44], including work that focuses specifically on the *screening* problem discussed here [22, 23, 27, 32, 42]. Interesting ways to combine ML and H classifiers have also been proposed, for example by building ML classifiers that can also include crowd votes as features, whose purpose is not only that of providing better classification but also of weighing the value (vs the cost) of obtaining additional crowd votes [17]. Other hybrid approaches ask the crowd to extract interesting patterns to be then fed to algorithms for classification, as opposed to relying on ML to do this [3, 35].

The approach we propose leverages the information provided by each kind of classifier (machine and human) to improve the effectiveness of the other kind so that they can be stronger together. The algorithm is based on a probabilistic model that adapts to the specific characteristics of each item, screening filter, workers' accuracy, and ML classifier available to identify *what* to ask next for each item (which filter to test) and *if* we should stop polling the crowd for a given item, either because we reached a decision or because we realize we cannot do so cheaply and confidently (that is, the problem of classifying that item is too hard for the crowd-machine ensemble). The latter point is particularly important: there may be items, filters, or classification problems in general where it is questionable whether machines and/or the crowd can classify with acceptable accuracy. If the algorithm can identify this subset of items and filters - or (item,filter) pairs - early and avoid spending money on them, then we know we can confidently try a crowd or hybrid approach as it will not waste money.

In essence, the approach - which we call *hybrid shortest run* (HSR) - works by considering the output of machine classifiers as a prior to the class probability for each item. This information is combined with an adaptive crowdsourcing algorithm that refines the class probability by polling the crowd on the (item,filter) pair that makes it more likely to reach a decision cheaply. In turn, classification information that is progressively obtained as we poll the crowd can be used to re-assess the performance of ML classifiers as well as create a model over the available classifiers for improved accuracy.

While many adaptive algorithms have been proposed (e.g., [4, 18, 30]), to the best of our knowledge this paper is the first to discuss hybrid classification in screening contexts, which has unique requirements and opportunities since, as we will see, the heart of the problem lies in identifying the filters (e.g., the exclusion criteria in SLR) that i) are most selective (screen out a large proportion of papers) and ii) that crowd or machines can classify accurately (can correctly determine if the exclusion criterion applies to a paper). If we can do so, we can focus on getting crowd votes for these filters first, leaving a smaller set of items for the filters the crowd finds hard to classify correctly or that do not have high selectivity.

We believe this paper is also the first to take a "black box" approach to hybrid classification: since we tackle finite pool problems, we do not assume we can develop an accurate, dedicated classifiers for each classification problem, although we may have available (possibly weak) classifiers from similar problems - in our SLR example these may be neural nets developed for other SLRs in the same or different fields or for other exclusion criteria. Rather, we identify how to leverage classifiers we are given and for which the accuracy for our task is unknown. We do this by a combination of (minimal) testing to filter out very poor classifiers and by combining the remaining ones to set a prior probability for whether a filter applies to an item. This helps us determine the most promising (item,filter) pairs for which asking a vote to the crowd has the highest probability of reaching a decision cheaply and confidently.

While it is not surprising that hybrid approaches have the potential to outperform human- or machine-only methods, the contribution we provide here lies in showing how we can cope with finite pools problems where potentially weak classifiers - and even classifiers with unknown accuracy for the problem at hand - can be leveraged effectively, even in contexts characterized by very demanding requirements in terms of accuracy (as is the case for SLRs) and in the presence of difficult problems where also the crowd has low accuracy. HSR always estimates the characteristics of the problem and the classifiers, and identifies which are the items and filters for which it can draw accurate conclusions cheaply, leaving the rest to expert classification.

We also study how the effect of incorporating classifiers (and ensembles of classifiers) into a crowdsourcing algorithm varies based on the different characteristics of the problem at hand (such as different selectivity of filters, accuracy of classifiers, correlation among base classifiers, amount of data available for testing classifiers and for training ensembles) and show why and under which conditions *ensembles* of classifiers do or do not provide benefits over "simply" using the best available classifier at hand.

2 RELATED WORK

2.1 Crowd Classification

This paper researches a field that is abundant in prior art. The problem of crowd classification has been studied for centuries, at least since the early work of the Marquis de Condorcet who, trying to make a case for democracy, proposed his *Jury Theorem*[1], stating that if each juror in a jury (or each citizen, when taking a binary decision) has an error rate lower than 0.5 and if *guilty* vs *innocent* votes are independent, larger juries reach more accurate results, and approach perfection as the jury grows.

More recently, with the opportunity offered by crowdsourcing on the one hand and the need for large labeled dataset to support ML tasks on the other, many researchers from the AI, database, and human computation communities investigated the problem in detail, both to identify algorithms and to study theoretical bounds. Initial algorithms were based on variations of (weighted) majority voting, where votes are counted differently based on the estimated worker's accuracy. Dawid and Skene [5] in the late nineties started a thread of research based on modeling and iteratively estimating workers' accuracy and class labels, by adopting variants of Expectation Maximization [6]. From there, many important and useful extensions have been proposed by Whitehill [44], Dong et al. [9], Li et al. [24], Liu et al. [25, 26] and many others. Relatively recent approaches based on spectral methods [20] and maximum entropy [45] have also been proposed, and belief propagation has been recently shown [31] to be optimal under certain assumptions.

Prior art has also discussed how to continuously assess the estimated value of asking for more votes from the crowd versus taking a classification decision with the votes as available. The approach is based on modeling the problem in terms of partially observable Markov Decision Processes [4, 18], where policies are progressively learned as votes become available.

2.2 Crowd Classification in multi-predicate problems

The set of classification problems we address here is that of *finite pool* classification [29]: we work on a finite set of items, and the classification criteria may be unique to the problem. SLRs are one case of such problems that has been thoroughly investigated by many researchers in recent years [22, 27, 28, 38].

Mortensen and colleagues crowdsourced paper screening [27] in four literature reviews, each with several criteria. They explore feasibility and costs of crowdsourcing, and they address the

[1] http://www.stat.berkeley.edu/ mossel/teach/ SocialChoiceNetworks10/ScribeAug31.pdf

problem by measuring workers agreement. Furthermore, they include a set of very interesting observations related to the importance of task design, as well as on the high degree of variability in workers's agreement from paper to paper and criteria to criteria (Fleiss' Kappa ranging from 0.5 to -0.03).

Krivosheev and colleagues [22] also present a model and strategies for crowdsourcing SLR. An interesting aspect of the model and approach here is that the authors model cost and loss (error) resulting from crowdsourcing task, attempt to estimate them at the start, and provide authors with a price/error trade-off that can be used to decide how much to invest in the task. They extend the work to the multi-filter screening case [21], also by introducing algorithms that efficiently query the crowd by focusing first on criteria that are more promising in terms of screening items correctly.

We borrow several concepts from this work and indeed we build over a slightly modified version of their algorithm. However, crowdsourced classification algorithms for us are to a large extent swappable components we leverage: we do not aim at improving crowd-only classification but rather to study how to efficiently combine ML and human classifiers, and specifically how to do so in a manner that is robust to weak classifiers to avoid compromising on accuracy and avoid spending money in classifications not likely to produce good results.

Multi-filter classification has been also widely studied in search and information retrieval contexts. Indeed, an instance of such problem is that of selecting which predicates to apply first when filtering tuples in a query. The seminal work in this area is by Hellerstein and Stonebraker, discussing the problem of ranking predicates in query plans [14].

More recently, Franklin et al. [12], Park and Widom [34] and Parameswaran et al [32, 33] focused on a similar problem but in the context of crowd-powered databases, while Anvur et al [1], Babu et al [2], Lan et al [23] presented *adaptive* extensions that tune the algorithm as the query progresses (see Deshpande et al [7] for a review). We differ in many ways with respect to the prior art in this area. First, again, we aim at combining ML classifiers with crowd. Furthermore, we follow an approach that makes initial estimates on the characteristics of the classification problem and then continuously revise the estimates, both in general and for each item to be classified, to identify an item and filter specific strategy to reduce errors and cost. Importantly, we identify the items on which we should give up trying, since it is more efficient to leave them for experts to screen.

Other prior work also addresses the issue of optimizations in terms of costs for obtaining labels and techniques to reduce cheating [11, 15, 16, 19, 37]. For example, Hirth and colleagues [16] recommend specific cheating detection and task validation algorithms based on the cost structure of the task. In this paper we do not optimize cheating detection or even task design to a large extent, in that those are orthogonal aspects with respect to the methods and optimizations discussed here.

2.3 Hybrid Classification

Just like crowd classifiers, ML classifiers are also ingredients of a solution we borrow from the state of the art. In the context of our problem the interesting issues are how to combine ML and humans, how to ensemble available classifiers, and how to balance the request cost of (crowd or expert) votes with the need of training and testing ML models.

Hybrid classification is an increasingly active area of research (see [39] for a recent survey). Many researchers study the problem in the context of clustering or entity resolution ([13, 40, 41, 43]). In general, many of these techniques operate by first classifying items automatically when ML classifiers are highly confident, and by then leveraging ML to shape the kind of task proposed to the crowd (e.g., identifying potential clusters or matching items to propose to the crowd [40, 41]). Another class of hybrid approaches, also extremely popular in industrial applications, are those where ML makes a proposal or a pre-filtering and humans confirm or refine. This happens in many fields, recently even in fashion [2].

Kamar and colleagues propose instead a promising approach where crowd features (votes, as well as potentially other aspects of the crowdsourcing process such as task completion times) and task features are combined into a broader set of features to be used to learn a model [17]. Researchers also explored using the crowd to extract features and patterns to then be leveraged by classifiers [3, 35].

While very interesting, these results are complementary to the work discussed here as we do not aim at developing a good base classifier but at leveraging effectively a set of arbitrarily weak classifiers we are given for the problem at hand. Perhaps the most closely related prior art is the work by Nguyen et al. [29], who adopt a clever mix of crowd+expert+machine learning with an active learning classifier, where papers to be labeled are iteratively chosen to minimize overall cost and loss, by comparing estimated loss of crowd classification versus expert classification. Our problem and approach differs, however, in that we focus on exploiting existing classifiers and in doing so for optimally selecting the (item,filter) pair to poll the crowd on. It is indeed in the optimization of which question to ask the crowd that lies the essence of the approach we propose.

3 MODEL, ASSUMPTIONS AND PROBLEM STATEMENT

We next define the problem we aim at solving and the model, which captures a few broad assumptions we make.

Specifically, we assume to have in input a set (I, F, C, L) where I is the set of items to classify (in our SLR example, these are papers to screen), F denotes the filters (paper exclusion criteria), C represents the set of ML or H classifiers, and $L = (k * FE + FI)/|I|$ is a loss function, modeled as a linear combination of false exclusions FE and false inclusions FI, which may carry a different relative weight k. The reason for the weight k is that in many contexts the two kinds of errors have very different implications. This is for example the case when screening is performed to reduce the number of cases to be brought to human attention, such as potential credit card fraud, tweets/posts potentially linked to criminal activities, or SLRs: screening out an item is often more "costly" than erroneously bringing the item to human attention.

The (possibly empty) set of given ML classifiers is assumed to be trained to receive an (item, filter) pair and output whether the filter applies to the item (e.g., receive a paper and an exclusion criterion and assess if the criterion applies to the paper). As mentioned, we do not discuss how to obtain classifiers (some examples are provided in the experiments section), as this is the subject of ample literature and it anyways depends on the problem domain. We do *not* necessarily assume that the classifiers are trained or even tested in the specific problem at hand (that is, for the specific filters and items under consideration), and therefore in general we do not even know their accuracy.

For human classifiers, we assume we have a set of generic workers of different and initially unknown accuracy and a set of high-accuracy but expensive experts.

Each classifier $c = \{cost, a(f), \rho(f, C)\} \in C$ is associated with the cost of asking one vote on an (item, filter) pair, with a filter-specific estimated accuracy (a 2x2 confusion matrix capturing probability of correct decisions for positive and negative labels), and with its *error correlation* ρ with other classifiers. The error correlation among two classifiers is the probability that both make an error given that one of them makes an error. Because we have no knowledge on the classifiers' accuracy, we conservatively model it as a uniform $Beta(1, 1)$ distribution for both positively and negatively labeled items (which means we do not even assume classifiers are better than random). If we do have additional information, this can be incorporated in the $Beta$. Consistently with

[2] https://multithreaded.stitchfix.com/blog/2016/03/29/hcomp1/

crowdsourcing literature, we also assume that crowd workers opinions are independent (that is, they make independent errors, given an item, filter and true label).

Each filter f is characterized by *difficulty* d_f and *power* θ_f. Difficulty reflects how easy it is for crowd workers to classify items correctly on that filter. Following Whitehill [44], we model difficulty as a real number $d_f \in [0, +\infty)$ that, given an expected accuracy α_w of a worker w, skews the accuracy as follows:

$$\alpha_{f,w} = 0.5 + (\alpha_w - 0.5) \cdot e^{-d_f} \tag{1}$$

As the difficulty d_f grows, $\alpha_{f,w}$ goes to 0.5, corresponding to random selection, which we consider to be the lowest accuracy level. The power $\theta_f$is simply the proportion of items to which filter f applies. Notice that we do not assume in this paper that power affects difficulty and we also do not assume different accuracies based on the true label of the items. Both difficulty and power are unknown and we assume no prior knowledge on them.

Given this model and assumptions, our goal is to identify an algorithm that can efficiently (in terms of cost) query the classifiers available and aggregate results while achieving the quality goals, stated in terms of loss as well as of the classical precision and recall measures.

4 BASELINE ALGORITHMS

We next describe baseline crowd and machine classification algorithms on which we base the construction of the hybrid approach. The crowd-based one is *shortest run* (SR) [21], a recently developed algorithm that has been shown to perform well for multi-filter screening. We summarize SR here to make the paper self-contained. There are other reasons why SR is particularly suited for the hybrid approach we propose, and we get back to this in the following.

4.1 Recap of Shortest Run

The Short Adaptive Multi-Run algorithm (Shortest run, SR for short) identifies and continuously updates an individual strategy for each item to be screened by identifying the shortest path to decision. In a nutshell, the idea borrows concepts from Partially Observable Markov Decision Processes (similarly to [4, 18]), by assessing based on the current state (that is, the knowledge accumulated thus far at the *global* level - power and difficulty for each filter - and at the *local* level - the votes on each specific item) which are the (item, filter) pairs to submit to the crowd for testing because they would be more likely to quickly (cheaply) lead to an accurate decision. SR may also decide to quit trying the crowd classification approach on an item if it believes that it would be too expensive to reach an accurate decision.

SR proceeds by performing what authors call a *baseline run* where a small set of items (usually 50) is screened with the standard approach of asking for labels on all filters (typically, 5 labels per item and filter) and classifying using a variant of Expectation Maximization. The result of the run is information on the power θ_f and difficulty d_f (and, correspondingly, workers' accuracy α_f) for each filter.

This information is used to obtain a prior estimate on the probability that the *next* vote v for each (item i ,filter f) will be a vote to screen out the item (that is, a vote that the filter applies to the item). We get an out vote if the item is out (with probability θ_f) and the worker answers correctly, or if the item is in (with probability $1 - \theta_f$ and the worker answers incorrectly:

$$P(v_{i,f} = OUT) = \alpha_f \cdot \theta_f + (1 - \alpha_f)(1 - \theta_f) \tag{2}$$

In addition, by applying Bayes, we know the probability of a filter f screening (i.e., $P(i \in OUT_f)$) or not screening ($P(i \in IN_f)$) an item i, once we obtain a label for the (i, f) pair.

$$P(i \in IN_f | v_{i,f}) = \frac{P(v_{i,f} | i \in IN_f) * (1 - \theta_f)}{P(v_{i,f})} \tag{3}$$

Because an item is screened out if at least one filter applies, then:

$$P(i \in OUT) = 1 - \prod_{f \in F} P(i \in IN_f) \tag{4}$$

An item is classified as out if $P(i \in OUT)$ is greater than a threshold $\overline{P_{out}}$. These formulas allow SR to estimate both what the next vote can be for a pair (i, f) and the impact each vote has on $P(i \in OUT)$. These estimates are updated as votes come in, where in Formula 3 the class probability, initially derived only from filter power and therefore equal for all unclassified items, is updated based on the votes obtained thus far for that item.

The above formulas can be easily extended to compute the probability that the next n votes for an (i, f) pair will be *in* or *out* votes. SR can in particular estimate the minimal number of votes $N_{i,f}^{min}$ in one direction (in or out) it needs to reach a decision, and the probability $P_{i,f}^{min}$ of getting such votes. As $N_{i,f}^{min}$ grows and its probability shrinks, SR may decide that crowd classification cannot be done efficiently, and quits trying (the stopping criteria are based on threshold which can be set as discussed in [21]). Intuitively, items that are left unclassified are essentially not filtered out, so they contribute to a higher loss, which has to be weighted against the price one would incur by insisting with the crowd. This in general is part of the same trade-off of how much users are willing to spend for unit of loss.

5 HYBRID SR ALGORITHM

While there is ample literature on how to generate a "good" ensemble of classifiers [8, 36], the prior art on the general problem of combining an existing set of ML classifiers $C = \{c\}$ is relatively less abundant [10]. We base the hybrid machine-crowd classification strategy on modifying SR. In the following, we first motivate why we start from SR and then introduce hybrid shortest run (HSR) by presenting, at each step, first the intuition and then the formalization.

The reason for starting from SR is that i) it was designed for multi-filter screening and has shown to perform better than baseline algorithms for crowd multi-predicate classification, ii) it has a per paper and per item probabilistic model that can leverage prior knowledge on items and filters, and ML classifiers can provide such knowledge, and iii) the algorithm can adapt to work with different test items T of different sizes. This is important, as test items can help us filter out ML classifiers with an expected accuracy lower than a threshold $\overline{a}$ (to be tuned as discussed later), and the more extensive set of crowd-classified items from the baseline can be used to a) assess independence among ML classifiers (which is necessary if we want to pool votes from ML classifiers with simple algorithms such as majority voting or Naive Bayes), and b) build an ensemble model out of the ML classifiers where the output of each ML classifier is a feature. This is similar to *stacking* [10] although we do not apply it over the training data used to build the individual classifiers (which we do not assume to have), but rather on the labels as estimated by the hybrid classification.

The Hybrid SR pseudo code is presented by Algorithm 1. The first step consists in filtering out classifiers that do not perform well for the case at hand. Specifically, we want to retain a classifier if we are confident it has an accuracy that is better than random. In principle, every contribution that is better than random helps, especially if it is independent from the other classifiers and if we weigh the vote by the classifier accuracy.

Algorithm 1: Hybrid SR Algorithm

```
Input: Input: $I, C, F, T, \overline{P_{out}}, \overline{P_{in}}, Budget$
Output: $CI = \{CI_{in}, CI_{out}\}$ #classified items
$CLF \longleftarrow$ Select classifiers based on T tests
$C_{meta} \longleftarrow$ Ensemble CLF
Prior, $\hat{\theta}^0 \longleftarrow$ run ensemble on I
$CI \longleftarrow \{\}, UI \longleftarrow I, k \longleftarrow 0$
$Votes^0, \hat{\alpha}^0 \longleftarrow$ Baseline run
#SRuns iterations
while $UI \neq \emptyset$ and $Budget \neq \emptyset$ do
    $k \longleftarrow k + 1$
    for $i \in UI$ do
        $f(i) \longleftarrow$ assign filter on i | Prior, Votes
        do check stop condition on i | Prior, Votes
    end
    $I^k \longleftarrow$ items with highest prob of classification
    for $i \in I^k$ do
        $v^k_{i,f} \longleftarrow$ collect a crowd vote on $f(i)$
        $Votes \longleftarrow Votes \cup \{v^k_{i,f}\}$
        if $P(i \in IN|Votes^k_i, Prior) > \overline{P_{in}}$ then
            $CI_{in} \longleftarrow CI_{in} \cup \{i\}$
            $UI \longleftarrow UI - \{i\}$
        end
        if $P(i \in OUT|Votes^k_i, Prior) > \overline{P_{out}}$ then
            $CI_{out} \longleftarrow CI_{out} \cup \{i\}$
            $UI \longleftarrow UI - \{i\}$
        end
    end
    $\hat{\theta}^k \longleftarrow$ update power
end
$CI^{difficult\_items} \longleftarrow$ label $UI$ as "IN items"
return $CI, CI^{difficult\_items}$
```

As commonly done in crowdsourcing, we assume we have access to a high accuracy test dataset T (obtained at a cost $C_T = ec * |T|$, that increases linearly by a factor ec with the number of test examples, where ec represents the cost for an ideal, expert screening). We then assign a prior probability distribution to each ML classifier and for each filter (ML classifiers may perform differently for different filters), reflecting our belief of how well the classifier performs on the problem at hand. As mentioned, in absence of additional information, we assume a $Beta(1, 1)$ uniform prior for a given filter. Other choices of prior are possible, assuming more likely that classifiers will have an accuracy above 0.5 and that accuracies close to 0 or 1 are very unlikely (for example, a Beta (3,2) has such characteristics). If we have a non negligible number of test cases the impact of this choice on the performance of the algorithm is relatively small. In the following for simplicity we assume a Beta (1,1) prior for all filters.

ML classifiers are tested with the gold dataset T (line 3), resulting in posterior probability $Beta(1{+}correct\_answers, 1{+}failed\_answers)$ which has a known distribution. We can, in particular, retain classifiers whose probability of being better than random is greater than a selection threshold sc, and this can be easily computed from the Beta distribution. Then we build an ensemble of ML classifiers and run it on a whole unclassified pool of items UI (lines 4,5).

Consistently with SR, we then perform a baseline run on B items (on all filters), both to estimate crowd accuracy on each filters and to get data for the next step (line 7).

This information is used to inform a prior $p_{i,f}^{mc}$ for each item and filter in Equation 3. In other words, while SR takes as prior for each filter f the proportion $1 - \theta_f$ of items classified as “in" for f and computes an overall prior, here we use the baseline ensemble output for each prior $p_{i,f}^{mc}$, where the class probability is the confidence that the filter applies. Equation 3 therefore becomes

$$P(i \in IN_f | v_{i,f}) = \frac{P(v_{i,f} | i \in IN_f) * (1 - p_{i,f}^{mc})}{P(v_{i,f})} \tag{5}$$

Notice that the prior has three effects: 1) it concurs to determine the classification probability; 2) it also affects which (item, filter) pairs we pick next for obtaining the crowd vote, since Equation 3 also affects $N_{i,f}^{min}$; 3) it has an impact on the stop condition (line 13), i. e., whether to stop to iterate over an item due to its classification difficulty or to continue. Once the (i, f) pairs to query next have determined, HSR proceeds to obtain the vote and iterates as per the SR algorithm (line 16).

We now present a series of experiments that, besides assessing the validity of HSR, help us understand its robustness as parameters of the problem and of the algorithm change. Besides comparing it with crowd-only classification, we also compare with ML-only classification. For this baseline comparison we leverage a Naive Bayes classifier, where machines are considered to be independent and where their votes are weighted by the estimated classifiers accuracy $A_c = \{a_c\}$ (by means of tests T). The ensemble is therefore defined as follows:

$$Class(l \in \{0, 1\}) \propto \prod_{c \in C} a_c^{\delta(l, l_c)} \cdot (1 - a_c)^{1 - \delta(l, l_c)} \tag{6}$$

where $\delta(l, l_c) = \begin{cases} 1 & l = l_c \\ 0 & l \neq l_c \end{cases}$ is the Kronecker delta function[3], and l_c is a label from a classifier c.

6 ANALYSIS AND EXPERIMENTS

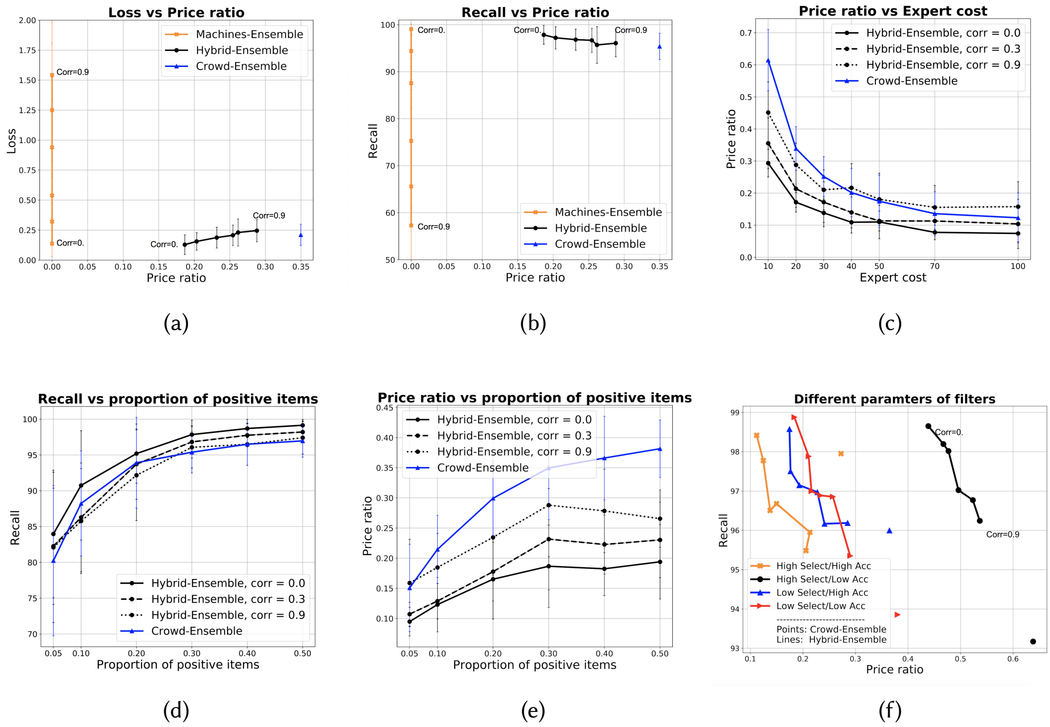
The approach to analysis is based on both simulations and crowdsourcing experiments. The simulations are interesting because they allow us to understand the behavior of the approach under very different conditions, and assess the impact of the variation of each parameter, be it a parameter that describes the nature of the problem (such as the filters' power and difficulty) or a parameter of the algorithm, such as the classification threshold $\overline{P_{out}}$.

The crowdsourcing experiments allow us to get actual values of parameters for real scenarios and assess validity of results in that context - besides understanding a set of nuances important in the setup of crowdsourcing tasks. We also leverage the data to build classifiers using commonly available techniques and use their accuracies to get a feel for the kind of accuracy we can achieve for the problem at hand.

[3] https://en.wikipedia.org/wiki/Kronecker_delta

Metrics. In the experiments there are three metrics we want to assess:

(1) The *loss*, computed as defined earlier, that quantifies our error weighted by how "severe" false exclusions are considered (we can similarly use F_β for this, though we choose loss in line with previous literature and also as it is more intuitive).
(2) The *recall*, which is as usual defined as *true inclusions / (true inclusions + false exclusions)*
(3) The relative cost of crowdsourcing, defined as the *price ratio* of the cost of crowd classification versus expert classification.



(a) (b) (c)

(d) (e) (f)

Fig. 1. Analysis of the hybrid algorithm as key parameters vary. Settings and description are provided in the text. Crowd-Ensemble refers to the SR algorithm, while Hybrid-ensemble is HSR.

The rationale for using price ratio as a metric when comparing algorithms is as follows: we consider that the price of crowdsourcing is not just represented by the total number CV of crowd votes asked: In fact, for each false inclusion FI (for each item the crowd fails to screen out) we need to incur in the expert screening cost, as we are leaving the item on the expert's desk. If we fail to account for this cost we overestimate the performance of the algorithm (incidentally, this is true for SR as well, although this is not discussed in the original paper and price ratio is not used there as a metric). We therefore compute the crowd cost as $CC = CV + FI * ec$[4] and compare it with what the total expert cost $EC = |I| \cdot ec$ (recall that I denotes the set of items to be classified) would be

[4] There are cases where the "expert cost" cannot be easily computed as the cost of expert labor: for example, a quasi-exact medical screening test may present practical challenges and even risks. In these cases, either domain experts can map these considerations into an "expert cost" metric, or the analysis needs to focus on precision, not price ratio.

if experts did all the classification. The price ratio is then CC/EC. Notice that considering price ratio and loss also incorporates precision: lack of precision (false inclusions) means higher loss and higher classification cost (higher price ratio). For this reason we do not plot precision in the following.

To describe the results, we first describe the baseline parameter settings for the experiments, and then show how results changes as we vary each parameter.

Baseline experiment. The baseline experiment settings consists of a problem where we screen items via 4 filters, and where 30% of the items survive the screening - if we classify them correctly. We simulate 10 ML classifiers with accuracy randomly selected from a 0.5-0.95 range and on which the algorithm assumes no prior knowledge. We screen them with T=50 tests, work the math with the Beta distribution as described earlier, and keep the classifiers with 0.95 probability of having an accuracy greater than 0.5. The classification threshold $\overline{P_{out}}$ is set at 0.99, the loss ratio k in the loss function is set to 10, and the expert classification cost ec is set to 20 times the crowd label cost. This latter value is estimated from [27] for the case of SLRs. Notice that expert screening cost is *per item*, while the unit label cost is *per label* on an (item,filter) pair. We stress again that users only have to set parameters corresponding to their requirements in terms of loss function, and the algorithm works out the rest, including the expected price ratios and recalls corresponding to given thresholds $\overline{P_{out}}$.

In Figure 1a we compare the results of applying machine only (ensembled with Naive Bayes), crowd (with SR), and HSR to simulations of classifications for 1000 items. The charts plot the mean loss by price ratio per item, obtained by averaging the results of 50 iterations. Vertical bars denote standard deviation. The different dots of the machines and hybrid curves denote different values for *error* correlation among machines, simulated at 0, 0.2, 0.3, 0.5, 0.7, 0.9. As we can see (Figure 1a), for a similar loss level, hybrid algorithms significantly outperform the crowd in terms of both price ratio and loss. Not surprisingly, price ratio and loss worsen as error correlation increases, as ML classifiers tend to agree when they make mistakes. However, even at high correlations we maintain performances that are superior to those of crowd-only. As we do not consider crowd correlation, for the crowd we just have a point in the chart. ML-only classification is competitive only with independent and accurate classifiers, and their performance quickly deteriorates even at small level of error correlation, as we do not assume we have strong classifiers. In the case of SLRs, with expert accuracy in the 95-98% range [21, 27] and assuming that the items that survive the screening are in the 10-30% range, the loss lies within 0.04 and 0.18 (with the settings of the experiment being closer to the 0.18 mark, slightly better than the result of HSR but a much higher cost).

Similarly, Figure 1b shows the recall vs the price ratio. Recall is independent of the loss parameter k and gives us a direct indication of our capacity to identify the items that pass the filters for a given monetary investment. Notice that we disregard here the cost of obtaining the base classifiers, which, if they are built from scratch for this specific SLR, needs to be factored in when estimating price and assessing the best strategy. On the other hand, if built for the specific problem they are also likely to have higher accuracy. We do not discuss this further as again building the base classifier is orthogonal to our goal here. We also observe that recall is high "by design" in that we fix the threshold $\overline{P_{out}}$ for considering an item as screened.

Effect of changes to the parameters. We now explore how results changes as we change either parameters that describe the nature of the problem (e.g., filter power, crowd accuracy, or loss and expert cost ratios) and the behavior of the algorithm (e.g., decisions on how many test data to obtain, or thresholds to set). Here we focus on a few interesting variations, and refer the reader to our GitHub repository[5] for additional details, data and source code. Notice that HSR is *adaptive*,

[5] https://github.com/Evgeneus/crowd-machine-collaboration-for-item-screening

that is, it estimates the parameters of the problem and changes its behavior accordingly. For this reason, we expect the recall to remain high, while price ratio may change.

Fig 1c plots how the price ratio improves with the expert cost *ec*. Recall (not plotted here) is essentially constant. For example, for a 0.5 correlation, it stays in the .96-.97 range for all *ec*. The lines flatten for high *ec* and asymptotically reach the minimal price ratio, which is the proportion of false inclusions.

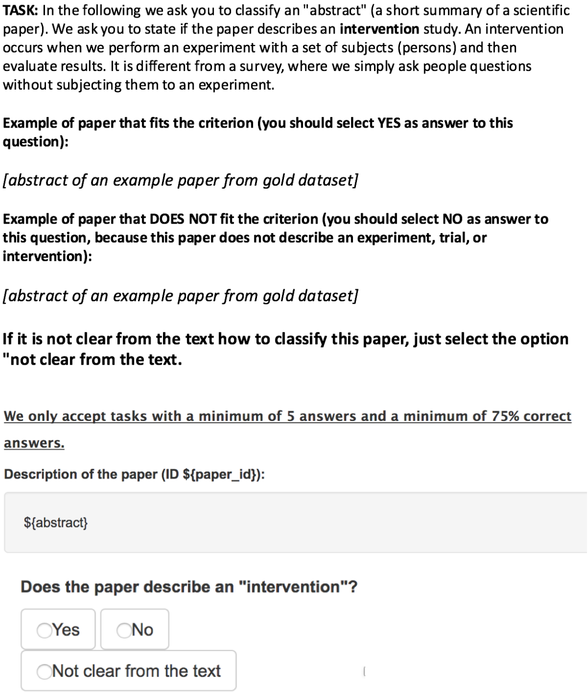
Figures 1d and 1e show instead how recall and price ratio change with the proportion of items that should pass the screening (positive items). Specifically, in the chart we show the behavior when the power of one of the four filters grows. The figures show that the performances worsen in high power situations. This is because high powers mean a prior $P(i \in OUT)$ that is very close to the $\overline{P_{out}}$ threshold, and as such it is more sensitive to errors from the crowd in the first few votes. This is not a problem of HSR per se but is inherited from SR. To correct it, it is sufficient to "tone down" the prior when the power estimate is too high. Indeed, we observed experimentally that this can be achieved by structurally underestimating power by approximately 20% of its value.

Finally, Figure 1f shows how recall varies in the presence of one criteria that heavily differs from the other in terms of difficulty (accuracy) and power (selectivity of filters). As expected, recall remains constant because the algorithm adapts to the characteristics of the problem. Price ratio worsens as we go from an easy and powerful criteria to the most difficult case of low power, low accuracy filter where we have many incorrect votes, and even correct ones do not help us much because the screening power of this filter is low, which means we anyways need to query the other filters. HSR results are robust to variations in the number of tests and in the confidence thresholds for keeping a ML classifier (not shown). Assuming a set of classifiers in the 0.3-0.7 range instead of 0.5-0.95 cuts approximately in half the gain with respect to SR.

Experimental data and experiment design. We experimented both by using existing datasets and by running crowdsourcing experiments. In all cases, data and code for the algorithms are available at the same url, along with other charts describing variations of behaviors with parameter values which we skip here as we find them less informative.

We first took an existing SLR dataset by Wallace and colleagues [27]. This includes over 20000 crowd votes over 4000 papers, with four filters. We also run a set of experiments on AMT and CrowdFlower, with different designs, but specifically borrowing the design from [21] for a fair comparison, and eventually built our own platform on top of AMT for increased flexibility and for being able to plug in the adaptive algorithms into the task assignment logic[6]. Specifically, the basic design included classifications of 100 papers for an SLR in the social informatics field, with three filters of different difficulty (one of them very difficult, with a worker accuracy barely above 0.5). The filters were: "does the paper describe a study on 65+ older adults", "does the paper describe a study that uses technology", and "does the paper describe an *intervention* type study" (the difficult one). The task proceeded as depicted in Figure 2. Workers with historical overall AMT accuracy over 70% were invited to participate. A worker would see a description of one criterion, along with a positive and a negative example. They would then proceed to labeling papers based on that criterion. The choice of a per-filter approach (asking for labels on many items for one filter) as opposed to per-item, is because it takes time to properly explain and understand a criterion (for example, explaining what an intervention is may be tricky and requires the worker to go through positive and negative examples carefully). We screen workers with two test questions and consider the labels from the remaining workers. After initial experiments to estimate the time taken by workers, we tuned parameters to arrive at a pay rate of 10USD/hour. We obtained 10 labels for each item and filter for a total of 3000 crowd votes from 147 workers.

[6] http://jointresearch.net/



TASK: In the following we ask you to classify an "abstract" (a short summary of a scientific paper). We ask you to state if the paper describes an **intervention** study. An intervention occurs when we perform an experiment with a set of subjects (persons) and then evaluate results. It is different from a survey, where we simply ask people questions without subjecting them to an experiment.

Example of paper that fits the criterion (you should select YES as answer to this question):

[abstract of an example paper from gold dataset]

Example of paper that DOES NOT fit the criterion (you should select NO as answer to this question, because this paper does not describe an experiment, trial, or intervention):

[abstract of an example paper from gold dataset]

If it is not clear from the text how to classify this paper, just select the option "not clear from the text.

We only accept tasks with a minimum of 5 answers and a minimum of 75% correct answers.

Description of the paper (ID ${paper_id}):

${abstract}

Does the paper describe an "intervention"?

○Yes ○No

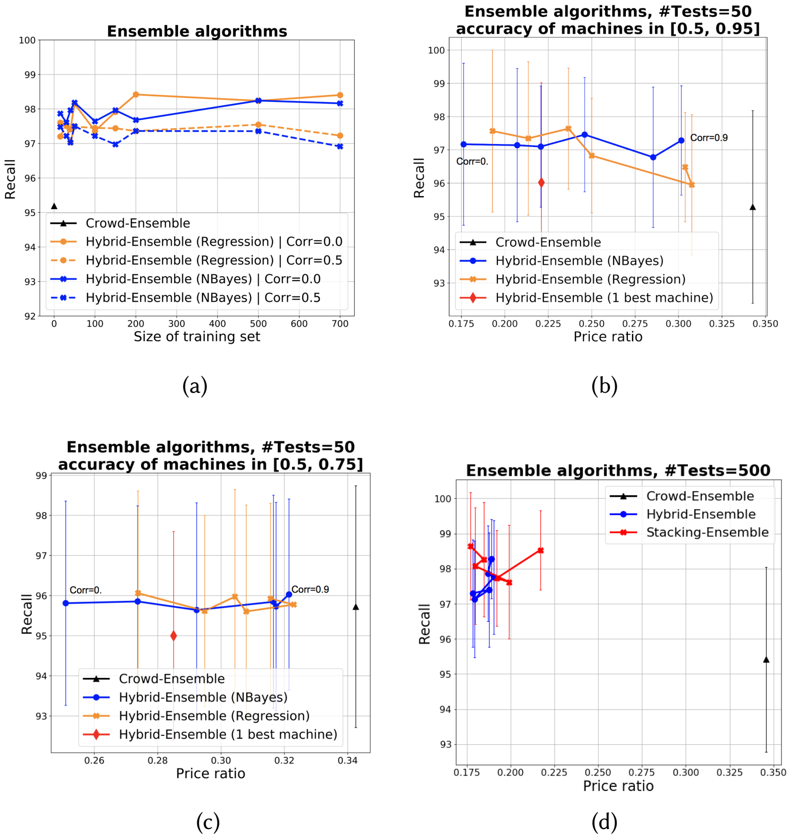
○Not clear from the text

Fig. 2. Classification task for SLR (settings borrowed from [21])

We then built classifiers for each filter using a variety of techniques (from KNN to random forest, variations of naive Bayes and logistic regression and others, with different kinds of document representations) and different sizes of training data to get realistic information on classifier accuracies and correlations. We obtained classifier accuracies in the 0.5-0.8 range, correlations in the 0.2-0.9 range, and crowd accuracy in the 0.55-0.8 range. In this case, classifiers were obtained by training on the same SLR data using 20% of the data, where the training set is obtained via crowdsourcing, so the savings for the experiment refers to the remaining portion of the items to be screened.

We then use these parameters obtained from experiments to fuel various experiments and repetition. We show the representative result in Figure 3d where the blue line represent HSR. We ran experiments with different combinations of base classifiers, having different correlations among them as usual represented by dots in the line. Notice that the standard deviation is very high in proportion to the scale of the chart, though we can see the impact of the hybrid approach. Higher correlations (moving from top to bottom) lead to slightly lower recall. Incidentally, we also report that, consistently with expectations, errors made by HSR and by us (the experts, in this case) were about the same: half the time we disagreed with the crowd, either the crowd was right or the decision was questionable.

Stacking classifiers. In the experiments above we naively included ML classifiers regardless of their error correlation. However, if two or more classifiers predict the same class (and especially if they make the same errors) then it does not help to pool their predictions - worse than that, when they make mistakes they make so by consensus thereby increasing our confidence in the incorrect prediction. There are essentially two approaches to deal with this: the first is to filter out highly correlated classifiers. If we denote with c^j_{err} the event corresponding to classifier c^j making



(a) (b) (c) (d)

Fig. 3. Experimental results for basic and stacking models. (a): recall for SR (black), basic HSR (blue) and HSR where classifiers are combined with regression (yellow) as the size of the training set used for the regression model varies; (b) and (c): recall vs price ratio for different distribution of classifiers accuracies; (d) results based on parameters obtained from AMT experiments. The dots in figs b,c,d present the correlation level of machine classifiers. Corr is [0, 0.2, 0.3, 0.5, 0.7, 0.9] from left to right.

an error, then we look to exclude classifiers where $P(c^j_{err}|c^k_{err})$ is high, and particularly above 0.5, otherwise we reinforce the error of c^k. This can be done again in the test phase similarly to what we do for estimating accuracy, for example again by assigning a Beta prior to this conditional probability. However, since errors might be rare, estimating error correlation requires more data points. If the initial (expert-provided) gold dataset is small, then we can proceed with HSR to get data points, initially with one or few ML classifiers, and then add more as we are confident of their low correlation.

However, we can also *stack* the ML classifiers by learning a model, and let this model filter or downweight correlated classifiers. Figure 3(a-c) show the performance of a logistic regression model built over the base ML classifiers and leveraged in HSR, compared with base HSR as from the previous section (blue) and with crowd-only ensemble (SR). The plot assumes 5 ML classifiers with accuracies as before, and a varying size of training dataset. In theory, the regression naturally copes with correlation and in the presence of a sufficiently large training set, it identifies how to effectively combine classifiers. As above, the training set can be progressively obtained via SR, and we can use base classifiers to identify a dataset that has higher possibility to be balanced, to get a better training set.

The charts show that in practice stacking with regression offers no improvement with respect to the naive aggregation of classifiers, and improvements are also limited with respect to taking the single best classifier in the set as opposed to an ensemble (but notice that SR takes us already to 0.95 recall, so improving from that is challenging).

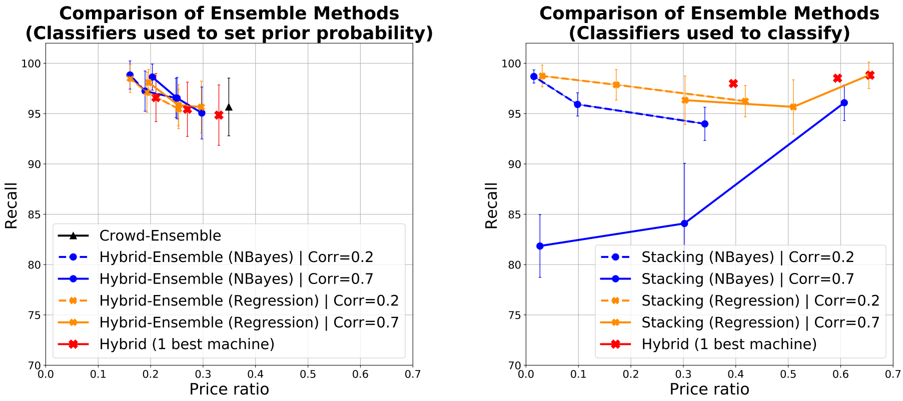
We found this result surprising, although consistent with the findings briefly mentioned by [10], where however the problem is not discussed besides the mentioning of this observation. To get clarity on this matter we investigated it further. Specifically, besides experimenting with classifiers of varying accuracies and correlation, we engaged to understand if the lack of effect is due to the specific way we use the output of classifiers - that is, to set prior probabilities (which in turn determine how to query the crowd for that item based on the HSR algorithm) as opposed to directly take a classification decision.

The results are shown in Figure 4. Figures a and b respectively show the performance of the ensembles when used as prior (Figure 4a) and when used directly to take final classification decisions, without resorting to the crowd (Figure 4b). The different dots denote accuracy of the base classifiers, with recall and price ratio improving with accuracy as expected (with one interesting exception discussed later). The two figures are shown in the same scale to facilitate comparison. We adopt the price ratio metric also for Figure 4b since classification errors mean that we incorrectly leave items for experts to classify, and this incurs in an unnecessary cost with respect to classifiers with perfect precision. Figure 4b shows that when we use the ensemble to make predictions, in low correlation conditions both Naive Bayes (NB) and regression ensembles are superior to the single best classifier especially in conditions of higher accuracy of the base classifiers (the difference in recall is small and within a standard deviation, while the benefit in terms of price ratio is high). With high correlation, regression ensemble and best classifiers offer similar performances, while NB drops significantly in recall, our main target metric. This is not surprising since in conditions of high correlation, ensembles with NB essentially reinforce the errors. Indeed, as accuracy of base classifiers increases so does our confidence in their vote (and in the ensemble). Specifically, often this confidence will exceed our confidence threshold $\overline{P_{out}}$ for classifying items as out directly, which explains the drop in recall for the NB ensemble as accuracy increases. In conditions of lower accuracy we take NB ensemble's predictions more cautiously, which explains the better recall (at the expense of price ratio). The benefit of regression however only manifests themselves once we have sufficient training data - in our experiments we started to observe a difference starting from training sets of 400 items.

When we leverage ensembles as prior (Figure 4a) as per HSR algorithm, these differences smooth out considerably and the points in the charts become closer. In HSR we never classify based on machines only, as we ask at least one vote from the crowd, which helps avoid such drops in recall. Ensembles improve over crowd-only approaches as discussed, even with weak classifiers, but because taking classifiers' results as prior is conservative and because classification threshold are high, recall errors from classifiers have a chance to be corrected by the crowd.

7 CONCLUSIONS

The main result of this paper is a method for multi-filter classification that combines ML and human classifiers to achieve high level of classification accuracy for unit of cost. We believe the main benefit lies in the ability of the algorithm to be robust to both the characteristics of the problem (such as filters that are hard for the crowd to classify) and to weak ML classifiers. In both cases, the



(a) (b)

Fig. 4. Comparison of NB and regression ensembles when used as prior (left) and to classify items directly (right). The dots in charts represent different accuracy range of machine classifiers ([0.65, 0.75], [0.75, 0.85], [0.85, 0.95]), increasing from right to left. Ensembles have been trained over a dataset of 500 items.

algorithm works out to leverage the filter, items, and ML classifiers (over specific filters and items) and builds models of classifiers to make the most of what can be screened automatically or with the crowd, leaving the rest for other classification methods (such as expert classification) without compromising on quality. Furthermore, the experimental results also helps us understand, in case we do have to build classifiers because we cannot transfer them from similar learning problems, what is the accuracy we should look for to get a benefit from hybrid classification. While we focused on hybrid algorithms we also uncovered improvements to multi-filter crowd classification algorithms such as the opportunity to smooth the prior in high-power filter scenarios.

Although the evaluation has focused on SLR we see the approach as applicable to many multi-filter screening problems. An extensive evaluation of the limits of the approach and the kinds of problems to which it applies is part of our current work. In addition, we see active learning for ensembles in finite pool contexts such as the one described here as an interesting research problem, which is complicated by the likely presence of highly imbalanced datasets and by needs that are often conflicting, that of leveraging the crowd efficiently to screen items out but to also get training data classified as in. Another interesting thread of analysis is to see if crowd workers can, besides labeling (item,filter) pairs, actually identify features that machines can then leverage for the problem at hand.

Finally, an interesting problem is the extent to which learning can be transfered across finite pool problems of the same kind (for example, different SLRs), especially in terms of identifying general approaches and methodologies that can be adopted in different domains.

ACKNOWLEDGEMENT

This work has received funding from the EU Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement 690692. It was also supported by the project "Evaluation and enhancement of social, economic and emotional wellbeing of older adults" under the agreement No.14.Z50.31.0029, Tomsk Polytechnic University.

REFERENCES

[1] Ron Avnur and Joseph M. Hellerstein. 2000. Eddies: Continuously Adaptive Query Processing. *SIGMOD Rec.* 29, 2 (May 2000), 261–272. https://doi.org/10.1145/335191.335420

[2] Shivnath Babu, Rajeev Motwani, Kamesh Munagala, Itaru Nishizawa, and Jennifer Widom. 2004. Adaptive Ordering of Pipelined Stream Filters. In *Proceedings of ACM SIGMOD*. ACM.

[3] Justin Cheng and Michael S. Bernstein. 2015. Flock: Hybrid Crowd-Machine Learning Classifiers. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work &#38; Social Computing (CSCW '15)*. ACM, New York, NY, USA, 600–611. https://doi.org/10.1145/2675133.2675214

[4] Peng Dai, Christopher H. Lin, Mausam, and Daniel S. Weld. 2013. POMDP-based Control of Workflows for Crowdsourcing. *Artif. Intell.* 202, 1 (Sept. 2013), 52–85. https://doi.org/10.1016/j.artint.2013.06.002

[5] A. P. Dawid and A. M. Skene. 1979. Maximum Likelihood Estimation of Observer Error-Rates Using the EM Algorithm. *Journal of the Royal Statistical Society. Series C Applied Statistics* 28, 1 (1979).

[6] A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)* 39, 1 (1977).

[7] Amol Deshpande, Zachary Ives, and Vijayshankar Raman. 2007. Adaptive Query Processing. *Foundations and TrendsÂő in Databases* 1, 1 (2007), 1–140. https://doi.org/10.1561/1900000001

[8] Thomas G Dietterich. 2000. An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization. *Machine Learning* 40, 2 (2000).

[9] Xin Luna Dong, Laure Berti-Equille, and Divesh Srivastava. 2013. Data Fusion : Resolving Conflicts from Multiple Sources. In *Procs of WAIM2013*. Springer. https://doi.org/10.1007/978-3-642-36257-6

[10] Saso Džeroski and Bernard Ženko. 2004. Is Combining Classifiers with Stacking Better than Selecting the Best One? *Machine Learning* 54, 3 (01 Mar 2004), 255–273.

[11] Carsten Eickhoff and Arjen P de Vries. 2013. Increasing cheat robustness of crowdsourcing tasks. *Information retrieval* 16, 2 (2013), 121–137.

[12] Michael J. Franklin, Donald Kossmann, Tim Kraska, Sukriti Ramesh, and Reynold Xin. 2011. CrowdDB: Answering Queries with Crowdsourcing. In *Proceedings of ACM SIGMOD*. ACM.

[13] Ryan Gomes, Peter Welinder, Andreas Krause, and Pietro Perona. 2011. Crowdclustering. In *Procs of Nips 2011*.

[14] Joseph M. Hellerstein and Michael Stonebraker. 1993. Predicate Migration: Optimizing Queries with Expensive Predicates. In *Proceedings of ACM SIGMOD*. ACM.

[15] Matthias Hirth, Tobias Hoßfeld, and Phuoc Tran-Gia. 2011. Cost-optimal validation mechanisms and cheat-detection for crowdsourcing platforms. In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2011 Fifth International Conference on*. IEEE, 316–321.

[16] Matthias Hirth, Tobias Hoßfeld, and Phuoc Tran-Gia. 2013. Analyzing costs and accuracy of validation mechanisms for crowdsourcing platforms. *Mathematical and Computer Modelling* 57, 11 (2013), 2918–2932.

[17] Ece Kamar, Severin Hacker, and Eric Horvitz. 2012. Combining Human and Machine Intelligence in Large-scale Crowdsourcing. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '12)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 467–474.

[18] Ece Kamar, Ashish Kapoor, and Eric Horvitz. 2013. Lifelong learning for acquiring the wisdom of the crowd. In *IJCAI*.

[19] David R Karger, Sewoong Oh, and Devavrat Shah. 2011. Budget-optimal crowdsourcing using low-rank matrix approximations. In *Communication, Control, and Computing (Allerton), 2011 49th Annual Allerton Conference on*. IEEE, 284–291.

[20] David R Karger, Sewoong Oh, and Devavrat Shah. 2011. Iterative learning for reliable crowdsourcing systems. In *Advances in neural information processing systems*. 1953–1961.

[21] Evgeny Krivosheev, Boualem Benatallah, and Fabio Casati. 2018. Crowd-based Multi-predicate Screening of Papers in Literature Reviews. In *Proceedings of WWW2018*. International World Wide Web Conferences Steering Committee.

[22] Evgeny Krivosheev, Valentina Caforio, Boualem Benatallah, and Fabio Casati. 2017. Crowdsourcing Paper Screening in Systematic Literature Reviews. In *Procs of Hcomp2017*. AAAI.

[23] Doren Lan, Katherine Reed, Austin Shin, and Beth Trushkowsky. 2017. Dynamic Filter: Adaptive Query Processing with the Crowd. In *Procs of Hcomp2017*. AAAI.

[24] Hongwei Li, Bin Yu, and Dengyong Zhou. 2013. Error Rate Analysis of Labeling by Crowdsourcing. In *Procs of ICML2013*.

[25] Chao Liu and Yi Min Wang. 2012. TrueLabel + Confusions: A Spectrum of Probabilistic Models in Analyzing Multiple Ratings. In *Procs of ICML2012*. ICML.

[26] Qiang Liu, Alexander T Ihler, and Mark Steyvers. 2013. Scoring workers in crowdsourcing: How many control questions are enough?. In *Advances in Neural Information Processing Systems*. 1914–1922.

[27] Michael L. Mortensen, Gaelen P. Adam, Thomas A. Trikalinos, Tim Kraska, and Byron C. Wallace. 2016. An exploration of crowdsourcing citation screening for systematic reviews. *Research Synthesis Methods* (2016). RSM-02-2016-0006.R4.

[28] An Thanh Nguyen, Matthew Halpern, Byron C. Wallace, and Matthew Lease. 2015. Probabilistic Modeling for Crowdsourcing Partially-Subjective Ratings. In *Procs of HComp2015*. AAAI Publications.

[29] An T Nguyen, Byron C Wallace, and Matthew Lease. 2015. Combining Crowd and Expert Labels using Decision Theoretic Active Learning. *Proceedings of the 3rd AAAI Conference on Human Computation (HCOMP)* (2015), 120–129.

[30] Besmira Nushi, Adish Singla, Anja Gruenheid, Erfan Zamanian, Andreas Krause, and Donald Kossmann. 2015. Crowd Access Path Optimization: Diversity Matters. In *HCOMP*.

[31] Jungseul Ok, Sewoong Oh, Jinwoo Shin, and Yung Yi. 2016. Optimality of Belief Propagation for Crowdsourced Classification. In *Procs of ICML2016*.

[32] Aditya Parameswaran, Stephen Boyd, Hector Garcia-Molina, Ashish Gupta, Neoklis Polyzotis, and Jennifer Widom. 2014. Optimal crowd-powered rating and filtering algorithms. In *Proceedings of VLDB*. VLDB Endowment.

[33] Aditya Parameswaran, Hector Garcia-Molina, Hyunjung Park, Neoklis Polyzotis, Aditya Ramesh, and Jennifer Widom. 2012. CrowdScreen: Algorithms for Filtering Data with Humans. In *Proceedings of ACM SIGMOD*. ACM.

[34] Hyunjung Park and Jennifer Widom. 2013. Query Optimization over Crowdsourced Data. *Proc. VLDB Endow.* 6, 10 (Aug. 2013), 781–792. https://doi.org/10.14778/2536206.2536207

[35] Carlos Rodriguez, Florian Daniel, and Fabio Casati. 2014. Crowd-Based Mining of Reusable Process Model Patterns. In *Business Process Management*, Shazia Sadiq, Pnina Soffer, and Hagen Völzer (Eds.). Springer International Publishing, 51–66.

[36] Lior Rokach. 2010. Ensemble-based Classifiers. *Artif. Intell. Rev.* 33, 1-2 (Feb. 2010), 1–39. https://doi.org/10.1007/s10462-009-9124-7

[37] Padhraic Smyth, Usama Fayyad, Michael Burl, Pietro Perona, and Pierre Baldi. 1995. Inferring ground truth from subjective labelling of venus images. *Advances in neural information processing systems* 7 (1995), 1085–1092.

[38] Yalin Sun, Pengxiang Cheng, Shengwei Wang, Hao Lyu, Matthew Lease, Iain Marshall, and Byron C. Wallace. 2016. Crowdsourcing Information Extraction for Biomedical Systematic Reviews. In *4th AAAI Conference on Human Computation and Crowdsourcing (HCOMP): Works-in-Progress Track*. http://arxiv.org/abs/1609.01017 3 pages. arXiv:1609.01017.

[39] Jennifer Wortman Vaughan. 2017. *Making Better Use of the Crowd: How Crowdsourcing Can Advance Machine Learning Research.* Survey and Position Paper. Microsoft Research. Available at http://www.jennwv.com/projects/crowdtutorial.html.

[40] Norases Vesdapunt, Kedar Bellare, and Nilesh Dalvi. 2014. Crowdsourcing Algorithms for Entity Resolution. In *Proceedings of VLDB*. VLDB Endowment.

[41] Ramya Korlakai Vinayak and Babak Hassibi. 2016. Crowdsourced clustering: Querying edges vs triangles. In *Procs of Nips 2016*.

[42] Byron C Wallace, A Noel-Storr, IJ Marshall, AM Cohen, NR Smalheiser, and J Thomas. 2017. Identifying reports of randomized controlled trials (RCTs) via a hybrid machine learning and crowdsourcing approach. *J Am Med Inform Assoc* (2017).

[43] Jiannan Wang, Tim Kraska, Michael J. Franklin, and Jianhua Feng. 2012. CrowdER: Crowdsourcing Entity Resolution. In *Proceedings of VLDB*. VLDB Endowment.

[44] Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier R Movellan, and Paul L Ruvolo. 2009. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. (2009), 2035–2043.

[45] D. Zhou, J. Platt, S. Basu, and Y. Mao. 2012. Learning from the wisdom of crowds by minimax entropy. In *Procs of Nips 2012*.

Received April 2018; revised July 2018; accepted September 2018