When in Doubt Ask the Crowd: Employing Crowdsourcing for Active Learning

Mihai Georgescu, Dang Duc Pham, Claudiu S. Firan, Ujwal Gadiraju, Wolfgang Nejdl
L3S Research Center, Leibniz Universität Hannover
Appelstr. 9a
30167 Hannover, Germany
{georgescu,pham,firan,gadiraju,nejdl}@l3s.de

ABSTRACT

Crowdsourcing has become ubiquitous in machine learning as a cost effective method to gather training labels. In this paper we examine the challenges that appear when employing crowdsourcing for active learning, in an integrated environment where an automatic method and human labelers work together towards improving their performance at a certain task. By using Active Learning techniques on crowd-labeled data, we optimize the performance of the automatic method towards better accuracy, while keeping the costs low by gathering data on demand. In order to verify our proposed methods, we apply them to the task of deduplication of publications in a digital library by examining metadata. We investigate the problems created by noisy labels produced by the crowd and explore methods to aggregate them. We analyze how different automatic methods are affected by the quantity and quality of the allocated resources as well as the instance selection strategies for each active learning round, aiming towards attaining a balance between cost and performance.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; I.2.6 [**Artificial Intelligence**]: Learning

Keywords

Human Computation, Crowdsourcing, Active Learning, Machine Learning

1. INTRODUCTION

Computers are adept at solving complex computational tasks, however there is a wide range of tasks at which humans are better. Tasks requiring subjective, perceptual, emotional or intellectual discretion capacities are some examples. The paradigm of social computation was developed specifically for solving these types of tasks. We propose an integrated system where humans and computers work together towards improving their performance on the task at hand. We tackle the problems that arise in building such a system and show that crowdsourcing coupled with machine learning can be a cost effective solution to alleviate specific problems where machines alone would fail, and employing only humans would prove to be too expensive.

We propose a framework where an automatic method incrementally learns from the crowd how to perform a certain task. Labels are provided by the crowd, on demand, for instances selected according to an active learning methodology. In each round the acquired labeled instances must maximize the information gain of the learner. We have included special provisions to take into consideration the noisy nature of crowd labels, and the diversity of workers. Furthermore, for instances with inconclusive aggregated labels, more labels from the workers should be collected, in order to build a reliable training set for the automatic methods. Each round of active learning has a corresponding cost that can be controlled with respect to the resources allocated to it. Only an optimal combination of selection strategy for new instances, allocation of resources, and crowd label aggregation will lead to a balance between cost and performance.

We propose solutions for the challenges raised by the proposed method of combining active learning and crowdsourcing. We experiment with different automatic methods, each learning from the crowd in its own distinct manner. For each round of active learning, resources are allocated to gather labels for new examples or for examples that persist with inconclusive aggregated labels. We carry out experiments to determine the optimal balance between the allocated resources and the performance increase of the trained automatic algorithm. Moreover, the instances for which new labels are gathered in each active learning round can be selected according to different strategies. We experiment with uncertainty, representativeness as well as a random selection strategy, confirming previous findings that a strategy which chooses instances that are representative of the entire pool of unlabeled data performs best. A challenge that arises when exploiting the crowd to complete micro tasks that pay a symbolical amount, is assessing the reliability of the work produced. However, with respect to the low costs concomitant with crowdsourced work when compared to employing experts, it can be beneficial.

Although the proposed methods can be applied to any domain, we focus on the special case of duplicate detection. Furthermore, we focus on the particular case where the entities are scientific publications. By duplicates we mean metadata documents that refer to the same real-world publication. As an example, consider an entity described

WIMS '14, June 02 - 04 2014, Thessaloniki, Greece

http://dx.doi.org/10.1145/2611040.2611047

by the following metadata: *title="As We May Think", author="Vannevar Bush", year="1945", book="The Atlantic"*. In different sources the fields might be represented differently, or in a source that builds the metadata collection using OCR there might be parsing errors or spelling errors, or some publications are present multiple times, each time with small differences in attribute values. For instance another representation of the same publication can be: *title="As we may think", author="V. Bush", journal="Atlantic Magazine"*. A publication search system needs to identify entities that match this criteria, in order simplify results lists, by grouping them at query time.

Different automatic methods can be used for computing labels for a pair of entities, using features learned by our proposed method from training data gathered using Amazon's Mechanical Turk[1] in an active learning environment. In each round we extend the available training set in order to optimize the performance of an automatic method towards better accuracy, guided by the training set and method settings learned in the previous rounds.

The contributions of this paper are: (i) we propose a novel integrated framework for active learning using crowd assigned labels; (ii) we identify the major challenges that can arise when deploying such a framework; (iii) we provide extensive experiments using various automatic methods that learn to perform a well-defined task by exploiting the the wisdom of the crowds; (iv) we envision further extensions that would make the proposed approach more robust.

The remainder of this paper is structured as follows. In Section 2 we review the state of the art in the fields we tackle. We propose our framework for active learning from the crowd in Section 3, followed by the dataset used to assess our methods in Section 4. We investigate the challenges that arise in implementing our proposed methods through extensive experiments in Section 5, and discuss the outcome of the experiments in Section 6. We finally draw conclusions and propose future directions for our work in Section 7.

2. RELATED WORK

Active learning [7] focuses on the costly acquisition of labels for the instances to be used for training in machine learning. [24] presents a survey of Active Learning and its applications. Cost sensitive active learning from noisy labelers was studied in [11], while in [25] it is shown that repeated labeling in a setting involving multiple and noisy labelers can lead to an improvement in data quality. As an alternative to getting more labels for the instances, the authors of [3] suggest that in cases of extreme class imbalance the labelers would be more useful if they would search for the instances of the missing class. We propose a framework for employing active learning and delegate the task of labeling to crowdsourcing, motivated by the cost efficiency. Furthermore, we examine how the quality of the multiple noisy labels and different selection strategies for new instances, affect the performance of the learner.

Human computation [30, 36] can be facilitated by means of crowdsourcing. Crowdsourcing either motivates participation with rewards or it relies on the altruism of the participants by bringing the larger purpose to light. Amazon's Mechanical Turk micro task marketplace is one of the most well known and used crowdsourcing platforms, and we thereby use it our experiments. The problems raised by the quality of the crowd data and aggregation of multiple crowd provided labels have been recognized and addressed in several papers. Some of the key problems are identified in [19]. The error-rate of workers can be evaluated using an EM algorithm [8], and biased workers, producing recoverable errors, can be separated from workers that always give erroneous answers [17]. By employing an EM algorithm in [22], the error-rates and the underlying hidden labels are estimated in the absence of a golden standard. The authors of [9] implement an unsupervised EM algorithm to jointly estimate worker accuracies and labels, whereas [33] implements unsupervised EM to jointly estimate labels and worker expertise while modeling example difficulty. To evaluate crowd annotations for natural language tasks, [27] implements a fully supervised estimation of worker confusion. [32] proposes unsupervised MAP estimation, jointly models example difficulty and annotator specific measurements of noise, expertise and bias. It has been shown that selective repetition of some micro tasks can improve data quality [25]. Strategies to collect high quality labels at a low cost for training retrieval models have been proposed in [35]. SQUARE [26] is a recently proposed benchmark for evaluating the quality of crowdsourced work and consensus. In our work the crowd is used for actively acquiring labels, thus the quality of the examples that will be used for learning is crucial. We acknowledge the noisy nature and the questionable reliability of crowd generated answers, and provide provisions for tackling this issue. After careful consideration, we use a majority voting strategy to aggregate the multiple crowd labels in most of our experiments. We also use the worker confidence measure described in a preliminary version of our methods [12] to expel the workers that introduce noise between active learning rounds.

By combining active learning with crowdsourcing, machine learning algorithms can be adaptively trained at a reduced cost. ZenCrowd [9] is a hybrid platform that combines algorithmic matching techniques and human intelligence to link entities using a probabilistic framework for the decision process and quality control. Active learning and crowdsourcing have been shown to be effective when used together for machine translations [2], computer vision tasks [6, 29], or to embed objects into an Euclidean space [28]. In the absence of a golden standard, our framework can actively learn from the crowd, in order to improve the performance of automatic algorithms, while determining the amount of work to be crowdsourced and tackling the quality issue.

The authors of [1] compare different types of new-instance sampling in an environment involving active learning from the crowd for a classification task. Bearing a fixed amount of monetary allocations, [18] evaluates the trade-off between asking for more labels and labeling more examples in active learning using Amazon Mechanical Turk. [34] addresses the question of what data to label and which annotator to use at the same time, by using probabilistic models and evaluating on simulated data. Unfortunately, there is currently no facility to select the desired annotators to use on Mechanical Turk. Incremental relabeling for active learning with noisy crowdsourced annotations has also been studied in [37]. We tackle most of the problems that have been identified in the literature and arise in the setting that we propose (active learning from the crowd). In addition, we solve the conundrum of either procuring more labels or labeling new ex-

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

amples, by retaining instances with inconclusive aggregated labels in the active learning loop.

As we test our proposed framework of actively learning from the crowd with respect to the task of finding duplicate records in a publications database, we promulgate some related work about duplicate detection. Duplicate detection has been referred to with different sobriquets: entity linkage [16] merge-purge [15], data matching [5, 10], deduplication [23], entity identification [21], reference reconciliation [13], or resolution [4]. It describes the process of finding similar records - entity descriptions - and deciding if two descriptions refer to the same real world entity. Entity resolution can be crowdsourced as shown previously in [31]. Our framework combines active learning with crowdsourcing for training an automatic method to perform a task. To showcase the capability of the framework through an analogous task, we demonstrate that it can tackle the particular task of finding duplicate entries in a collection of publications.

3. ACTIVE LEARNING FROM THE CROWD

We are employing an active learning technique to improve an *Automatic Method* so that its behavior fits better to how the crowd solves a particular task. Unlike an ordinary learner that is trained using a static training set, an active learner actively picks subsets of instances from the unlabeled data which, when labeled, will provide the highest information gain to the learner.

The general steps taken by our method are described in in Algorithm 1. We start with an empty list of instances that need to be labeled (*Candidates*), an empty training set (*Train*) and an empty set of workers that need to be excluded from the crowd label assignment process (*BadWorkers*). We use a *Selection Strategy* for guiding the active learning process, and a *Label Aggregation Strategy* to aggregate the crowd labels (CSL) and in the same time assess the worker performance (WQ). Two thresholds are used in the algorithm: the worker confidence threshold *WQThreshold* and the aggregated crowd label confidence threshold *CSLThreshold*. As an output of every active learning round the algorithm provides a training set for the Automatic Algorithm, *Train*, that will get better and better with each iteration, the workers that were identified to provide labels of unsatisfactory quality, *BadWorkers*, and the instances for which we need to get more labels in order to have a conclusive and confident aggregated crowd label, *Candidates*.

In consecutive rounds we repeat the same learning process: To the instances that still have an uncertain status, *Candidates*, we add a sample of instances provided by a candidate instances *Selection Strategy*. We prepare a batch of micro tasks to be solved by the crowd, blocking first the workers that were identified to be unreliable, *BadWorkers*. After the microtasks are solved by the crowd, we compute the aggregated crowd decisions and the worker confidence using the desired *Label Aggregation Strategy*. We monitor the worker confidence, and the workers that fall below a certain accountability threshold can be blocked so that they do not continue to introduce errors to our system. We identify the *High Confidence* set, the instances for which the confidence and agreement between workers indicate a clear label. We use this set to improve the performance of our Automatic Method. If we do not have a high agreement between the workers, or if the confidence in the aggregated crowd label is not high enough, then we need to get more opinions on these instances, by extending the number of assignments of the corresponding microtasks, and keeping them in the Candidates set. At the end of each round we extend the training set by acquiring labels that would increase the performance of the automatic method, and we identify workers that are not reliable and instances for which we need to get more labels from the crowd. We can stop the active learning process and the loop once we get a satisfactory performance using the training set built over the consecutive rounds. In the following sections we will point out the main features of each step in our algorithm, corresponding to different components of the framework.

Algorithm 1 Active Learning from the Crowd.

Input: Sample size: s instances per round
$AutomaticMethod$
$LabelAggregationStrategy$
$SelectionStrategy$
$WQThreshold$; $CSLThreshold$
Output: $Train$: Training set for the $AutomaticMethod$
$BadWorkers$: Low quality workers
$Candidates$: Instances to label

1: $Candidates = \varnothing$
2: $Train = \varnothing$
3: $BadWorkers = \varnothing$
4: Use default of common sense parameters for the $AutomaticMethod$
5: **loop**
6: Add to $Candidates$ a set of instances of size s chosen by the $SelectionStrategy$
7: Prepare a batch of micro tasks containing instances in $Candidates$
8: Post micro tasks on the Crowdscourcing marketplace excluding workers in $BadWorkers$
9: Retrieve labels from the Crowdsourcing marketplace
10: Compute the aggregated crowd labels CL and CSL and assess the worker confidences WQ using the desired $LabelAggregationStrategy$
11: Add workers with $WQ < WQThreshold$ to $BadWorkers$
12: Add instances with $CSL \geq CSLThreshold$ to $HighConfidence$
13: $Candidates = Candidates - HighConfidence$ (Keep the instances for which the confidence in the aggregated label is not strong enough for the next iteration in order to get more crowd labels)
14: $Train = Train + HighConfidence$ (Add the high confidence instances and their aggregated labels to the training set for the Automatic Method)
15: Retrain the Automatic Method with the new $Train$
16: **return** $Train$, $BadWorkers$, $Candidates$
17: **end loop**

3.1 Gathering Labels from the Crowd

In order to optimize the Automatic Method such that it provides results as close to reality, we use a crowdsourcing marketplace like Amazon's Mechanical Turk (MTurk), since it can provide labeled data for machine learning rapidly and at a low cost. Consequently, we delegate the task of assigning labels to MTurk workers. On MTurk a unit of work is called a *HIT*, Human Intelligence Task, and it can be done

in five minutes or less, for a monetary reward. Each HIT has an extendable maximum number of *assignments*. The data is sent to MTurk in batches. With each solved batch, the automatic method learns from the crowd how to do the task better. The automatic method thus improves; although there will always be some instances on which the label assignment is uncertain, this number will decrease, together with the need for input from the crowd.

3.2 Aggregation of Crowd Labels

Different *Label Aggregation Strategies* can be used for aggregating the crowd labels such as those proposed in [12]: majority voting, iterative estimation using an EM algorithm, boosting worker confidences when using the same EM algorithm, or different heuristics. The aggregation of different labels provided by the crowd workers can take into account features of the instances to be labeled, as well as worker features. Let us refer to the aggregated *Crowd Label* of an instance i as $CL(i) \in \{1, -1\}$. For the task of duplicates detection 1 stands for duplicates and -1 for non-duplicates. It can have an associated score, that is an indicator of the confidence we have in it being the true label, called *aggregated label confidence* or aggregated *Crowd Soft Label*, $CSL(i) \in [0, 1]$. The confidence we have in the label tells us if we need to get more labels for the instance, in order to strengthen the aggregated label. If the confidence surpasses a chosen threshold, *CSLThreshold*, then we do not need any more labels and we can use the aggregated label as it is in the training of the automatic algorithm. A simple measure can be the agreement between the workers providing the labels. The EM based methods provide a Soft Label that can serve this purpose. Furthermore, the confidence of the aggregated label can allow more advanced automatic methods that take into account label uncertainty when training in order to improve their performance.

To compute an aggregated crowd label, the difference in quality of labels provided by different workers has to be taken into account. Therefore, we have to assess the quality of the labels provided by each worker; we will further refer to this measure as worker confidence, WQ. Some workers perform better than others depending on their understanding of the task and their background and experience. The weight that a label has in the aggregation should be proportional to how good the worker providing it is. We want to penalize the votes coming from workers that do not provide good quality answers, and boost the weight of the answers of those workers that are good. Eventually, we build a database of workers that consistently underperform, the *BadWorkers*, and block them from taking part in our tasks. These are the workers with an evaluated WQ over one or multiple consecutive rounds that falls under the acceptable worker confidence threshold, *WQThreshold*. Eliminating the workers that are consistently providing low quality answers would reduce the noise and lead to better labels. There are also users that consistently provide the wrong answers, because they have malicious intentions. There are methods proposed in the literature, such as [17] to recover the answers of such workers. Furthermore, worker quality can also be dynamic, not static, depending on their evolution in time. Some workers might get better at the task as they build up experience, while other workers might get worse during a session of work because of boredom induced by repetitive activities. This temporal dimension of the involvement of the user can also be taken into account when assessing how good they are, in order to proportionally weigh their labels when aggregating all the available labels for an instance.

After careful consideration we decided to use majority voting for most of our experiments, as it provides a good trade-off between performance and computation effort. Nevertheless, in the experiments related to creating the $BadWorkers$ set, we employ aggregation strategies that provide a measure for the worker performance.

3.3 Candidate Instances Selection Strategies

In each round the instances that will get new labels from the crowd workers are selected using a *Selection Strategy*. The way these instances are selected influences how fast the Automatic Method learns. The selection strategy guides the active learner as it uses the existing model to actively pick subsets of instances from unlabeled data which, when labeled, will provide the highest information gain. The strategy should select the instances such that the learning process is sped up when compared to a random selection. We propose two different strategies for selecting instances to be labeled by the crowd:

- **Uncertainty Selection** We choose those instances for which our Automatic Method is most uncertain about the label assignment, depending on how this is computed.

- **Representative Selection** We choose instances such that they cover the entire spectrum of certainty in the assigned labels, depending on how the labels are computed by the Automatic Method.

3.4 Improving an Automatic Method

The automatic method produces an *Automatic Label* for an instance i, $AL(i) \in \{1, -1\}$ Furthermore the automatic methods can produce an *Automatic Soft Label*, $ASL(i) \in [0, 1]$, which serves as an indicator of the confidence the automatic algorithm has in its label assignment. This soft label assignment is used by the *Selection Strategy* to find instances that will provide the highest information gain to the automatic method when added to the training set.

As already mentioned in the introduction, although the methods proposed can be applied to any domain, we focus on the special case of duplicate entity detection, tackling the particular case where entities are scientific publications. Let us use the following notations:

e_i represents an entity, described as a set of attribute-value tuples:

$$e_i = \left\{(F_k, V^i_{F_k}))|F_k \in FieldNames\right\}$$

,where F_k denotes the field name and $V^i_{F_k}$ its value for entity e_i. The field name F_k belongs to a fixed set of possible field names $FieldNames$. $p_{i,j}$ denotes a pair of entities (e_i, e_j) that can be duplicates or not.

In this case the instances for which we need to get labels are pairs of publications, $p_{i,j}$. Thus, the automatic method produces an $AL(p_{i,j})$ being 1 for a duplicates pair and -1 for non-duplicates pair, and an $ASL(p_{i,j}) \in [0, 1]$.

We propose two types of automatic methods that can learn from the crowd labeled data obtained through the active learning acquisition process.

3.4.1 Duplicates Scorer

The **DuplicatesScorer** is introduced in [20] and used in [12] for the task of identifying duplicate publications. For a given pair of entities $p_{i,j}$, the algorithm computes an Automatic Soft Label $ASL(p_{i,j}) \in [0,1]$ as a variant of *ϵ-adjusted geometric mean* based on the parameters

$$DSParams = \{(F_k, W_{F_k}) | F_k \in FieldNames\}$$

,where $W_{F_k} \in [0,1]$ denotes the weight of the F_k field in the computation. The decision for the final label is taken by comparing the soft label with a chosen *threshold*. The automatic algorithm is designating $p_{i,j}$ as a duplicates pair, therefore $AL(p_{i,j}) = 1$, if $ASL(p_{i,j}) \geq threshold$, and as non duplicates, $AL(p_{i,j}) = -1$, otherwise.

When using it as an automatic method in the active learning framework we start with a common sense parameter choice to identify the initial items for which we get labels according to the selection strategy. In the subsequent consecutive rounds we learn new parameters for the DuplicatesScorer using the knowledge of the parameter choice identified in the previous round. Training this method is done by using an optimization algorithm that finds the parameters choice that maximizes the accuracy on the testing set. More details on how this automatic method can be trained according to different decision and optimization strategies are given in [12], and we briefly mention them here: (i) maximizing the accuracy when comparing the final label assignments for the crowd, CL, with the automatic labels, AL, or (ii) maximizing the correlation or minimizing the sum of errors or sum of log of errors when comparing the soft aggregated crowd labels, CSL, with the soft automatic labels , ASL. The measure of label uncertainty used by the *Selection Strategy* is the distance between the $ASL(p_{i,j})$ and the *threshold*. For this automatic method, training is equivalent to optimizing the $DSParams$ and the *threshold* such that the labels provided by it match those provided by the crowd.

3.4.2 Classifiers

We can employ a classifier using different features for assigning the labels $AL(p_{i,j})$ to the instances. For a given pair of entities $p_{i,j}$ the feature set used consists of the similarities between the different fields that characterize each entity.

$$Features(p_{i,j}) = \left\{ sim(V^i_{F_k}, V^j_{F_k}) | F_k \in FieldNames \right\}$$

As a similarity measure we propose to use either the Jaccard or the Needleman-Wunch similarity.

When using a classifier in the active learning framework, we start by choosing a random sample of instances for training. In the subsequent consecutive rounds we take into consideration the certainty of labels being assigned to instances in accordance with the selection strategy and the already existing training set, in order to select new instances and improve the performance of the classifier by building a better training set with each round.

4. DATASET

Without restricting the generality of our methods, we focus on the domain of digital libraries and present examples of scientific publications. The entities we deduplicate are represented by scientific publications, therefore as instances we consider pairs of publications labeled either as "duplicate" or "non-duplicate" pairs. The complete list of fields that a publication can have is *FieldNames*={ *Title, Subtitle, By, In, Type, Publisher, Organization, Abstract* }.

The *By* field is composed of person-related data like *Author*, *Editor*, *Contributor*. The *In* is composed of venue related data like *Book*, *Journal*, *Conference*, *Venue* and *Series*. The most discriminative field is of course the *Abstract* because this is distinct for each publication, but not all the publications in our dataset contain it. If all publications would contain the *Abstract* field (which is not usually the case), the task of identifying duplicates would be trivial, being reduced to comparing just this field for the publications in a pair. We have not used the publication year in our experiments since we found this to be mostly unreliable, considering that entries of the same publication could be associated with dates differing by as much as 2 years.

The dataset is composed of publication pairs, that can be labeled as duplicate or non-duplicate pairs. The publications come from 4 different sources: DBLP[2], CiteSeer[3], BibSonomy[4] and TIBKat[5], as presented in the scientific publications search engine *Freesearch*[6]. Having documents from multiple sources, with various data quality means that there will be duplicates, between sources, or even within the same noisy source, to be identified using the proposed methods.

4.1 Data Gathering

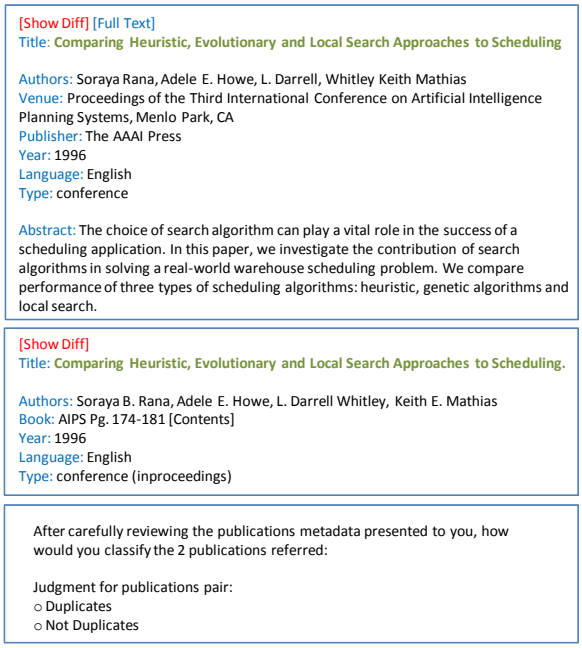


Figure 1: Example of a Mechanical Turk Task.

For label gathering we create HITs consisting of 5 publication pairs for which the workers have to assign the duplicate or non-duplicate labels. Detailed instructions are provided, and in addition examples help the workers to understand and solve the task better. Moreover, the worker can also see

[2] http://www.informatik.uni-trier.de/~ley/db/
[3] http://citeseerx.ist.psu.edu
[4] http://www.bibsonomy.org
[5] http://www.tib-hannover.de/de/die-tib/opendata/
[6] http://dblp.kbs.uni-hannover.de

the differences between the two publications, highlighted in different colors. Each HIT pays 5ct. to the worker that solves it. We start with 3 assignments, and if we need to get a more categorical decision from the workers for a pair we extend the HIT that contains it by adding more assignments. We present an example of how a publication pair is shown to the crowd workers in Figure 1.

The *Ground Truth* was manually labeled by 3 experts and is composed of 363 pairs, of which 101 are considered duplicates and 262 non-duplicates. The *Crowd Data* was labeled using Amazon Mechanical Turk, and it is used for improving the automatic methods by using the active learning label gathering strategy. It consists of 2070 pairs with at least 3 corresponding votes each, out of which 570 pairs have 7 votes each. According to the majority vote the 2070 pairs are split into 808 duplicate pairs and 1262 non-duplicate pairs.

We extended the dataset presented in a previous work [12] to include more votes for the pairs where the agreement was deemed to be poor. Hereafter, we briefly describe the label gathering strategy we used for extending the existing dataset. The data is gathered in 3 rounds, guided by a deployment of our algorithm with the Duplicates Scorer as an automatic method, and the uncertainty selection strategy. To test the hypothesis that with each round of active learning, our algorithm will exhibit an improvement and need less and less input from the crowd, the performance is observed over running it for 3 rounds. The accuracy of each run is presented in Table 1, along with the number of pairs included in the sample used for training. Additionally, the different weights for the fields reported by each run are presented. For Round 0, the starting point of the active learning process, we started with a common-sense parameter choice, in order to be able to compute initial scores and select a sample of pairs with an uncertain assignment. In the following rounds we always start optimizing from the parameters we learned in the previous round. The pairs for the forthcoming round are selected so that their score computed using the current weights is around the threshold of the current round.

The weights change between rounds, but remain close to the ones computed in the previous rounds. The learned threshold remains in the proximity of the common-sense threshold that we started with. The accuracy slightly increases, validating our hypothesis, that the parameters learned in consecutive rounds provide better performance. For the experiments on the optimal number of tasks to be solved in a round, we used the same parameters as in round 1 to get 500 more pairs, in order to ensure pairs are selected in the same way. Therefore, we extend the 570 pairs having 3 votes from [12], by gathering 4 more votes for each of them, collected 500 more pairs with 3 votes, and then in 2 consecutive rounds 1000 more pairs with just 3 votes.

To investigate any link between the importance of the attributes and their distribution in the dataset, in Figure 2 and Figure 3 we present a histogram of the number of pairs and field names for which a pair has values for either both, none, or only one field name, in the ground truth as well as in the crowdsourced data respectively.

The distribution for the Ground Truth is similar to that of the Crowd Data. Furthermore, we can see that all of the pairs have the *Title* field present. The other fields that appear in both publications of most pairs are *In* , *By* and *Type*. The *Subtitle* and *Organization* fields are in most cases empty for both publications in the pair. *Publisher* and *Abstract* are well distributed among the classes. Only a small proportion of pairs have the most discriminative field, the *Abstract*.

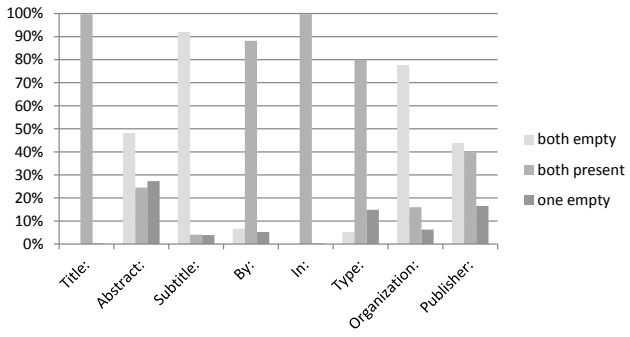


Figure 2: Analysis of attribute distribution in the Ground Truth data.

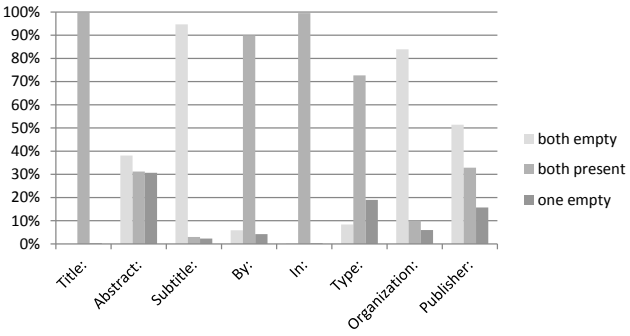


Figure 3: Analysis of attribute distribution in the Crowd data.

4.2 Agreement between Labelers

In this section we present statistics regarding the agreement between annotators in terms of Fleiss' Kappa and Krippendorf's Alpha. In section 6.1 we will discuss further about the correlation between the annotator agreement and the performance of our methods. In Table 2 we can see the number of instances that have as many assigned crowd labels, as well as the two indicators for measuring the agreement between annotators. We can clearly notice that the agreement between the expert assessors for the ground truth is the highest. On evaluating the agreement between crowd labelers on the ground truth, we notice that contrary to our expectations, introducing more labelers actually decreases their agreement. We do not have the same number of instances for each choice of number of labelers, but 3 assessors are clearly in more agreement than 5. In contrast to this, on evaluating the agreement of the crowd on all the data, we see that introducing more than 3 labelers actually increases the agreement, 5 workers resulting in more mutual agreement than just 3. Nevertheless, in this setting introducing more than 5 workers also has a detrimental effect on the agreement. The agreement of the crowd on the ground truth data is comparable to the agreement on the whole data, except in the case where we employ 3 workers.

When comparing the labels where the crowd labelers did not agree with one another and the ground truth we notice several patterns. Most significantly the crowd labelers assign the duplicate label to non-duplicate pairs, being influenced mainly by the title in order to produce their labels.

Table 1: Accuracy, number of pairs used for optimization, threshold and weights for DuplicatesScorer learned in consecutive Active Learning rounds in the data gathering process.

Rnd	Accuracy	Pairs	Thresh.	Abstract	Title	Subtitle	In	By	Type	Organization	Publisher
0	0.77027	-	0.75	0.5	1	0.5	1	1	0.5	0.5	0.5
1	0.78815	570	0.72	0.17	0.98	0.2	0.05	0.3	0.19	0.01	0.26
2	0.78981	500	0.73	0.55	0.97	0.1	0.27	0.37	0.21	0.0	0.01
3	0.79504	500	0.75	0.7	0.98	0.1	0.47	0.54	0.13	0.0	0.04

Table 2: According to the number of labels available(Labels), the number of instances in the dataset having at least that number of labels (Instances), and two agreement indicators between the labelers: Fleiss' κ (κ) and Krippendorf's α (α).

Labels	Instances	κ	α
Experts Agreement on GT			
3	362	0.827	0.827
Crowd Agreement on GT			
3	358	0.526	0.526
4	358	0.526	0.526
5	358	0.503	0.511
6	337	0.478	0.499
7	285	0.47	0.492
Crowd Agreement on all data			
3	2064	0.282	0.282
4	570	0.506	0.303
5	570	0.499	0.319
6	570	0.495	0.331
7	570	0.477	0.338

5. EXPERIMENTS

5.1 Accuracy for Different Methods and Aggregation Strategies

As a first experiment we assess the performance of the different Automatic Methods: classifiers and Duplicates Scorer. We used Majority Voting as a label aggregation strategy.

The chosen classifiers are Naive Bayes (*NB*), SVM (*SVM*) and Decision Tree (*DT*). We used their respective Weka[14] implementations: Naive Bayes, SMO and C4.5 (J48) with the default parameters. A pair of $p_{i,j}$ is characterized by 8 features: $sim(V^i_{F_k}, V^j_{F_k})$ where $F_k \in \{$*Title, Subtitle, By, In, Type, Publisher, Organization, Abstract*$\}$ and *sim* can be either the Jaccard similarity based on tokens or the Needleman-Wunch similarity at character level.

The parameters of the Duplicates Scorer(*DS*) are the weights of the fields describing the publications *Title, Subtitle, By, In, Type, Publisher, Organization, Abstract.* They are used to compute $ASL(p_{i,j})$ and the *threshold* for deciding if the pair is a duplicate or non-duplicate pair. As a learning process for the Duplicates Scorer we used the accuracy maximization optimization strategy.

In Figure 4 we report the accuracy for different settings: training on the crowd data and testing on the ground truth(*c-gt*), 10-fold cross-validation when training and testing on the ground truth (*gt-gt*) and 10-fold cross-validation training and testing on the crowd data (*c-c*).

We notice that in all the settings the Duplicates Scorer is surpassed by the classifiers, and among the classifiers, the decision tree seems to provide the best results. We also observe that the different settings for the training and testing data have different accuracies. We do not observe any remarkable differences between using the Needleman-Wunch or Jaccard as a similarity measure. The highest accuracy is obtained when training and testing on the ground truth data. This might be due to the high quality data, which is also reflected in the agreement statistics presented in Section 3.2. Training on the crowd data and evaluating on the ground truth also results in a good performance. The lowest performance is reported when training and testing on the crowd data. The lower performance obtained when using crowd data for testing might be due to its inherent noisy nature or lower quality.

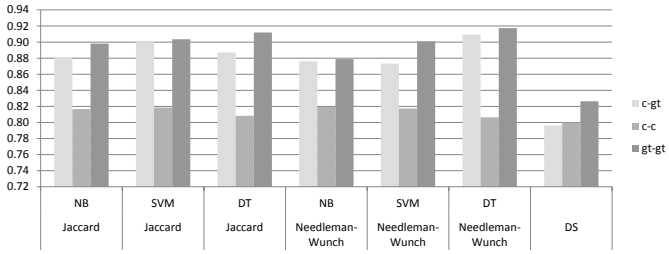


Figure 4: Accuracy of the different methods in various settings.

5.2 Attribute Selection

In order to determine which fields are more important for different automatic methods we used 3 attribute selection methods. The results are presented in Table 3. For the Duplicates Scorer we ran 'Leave one field out' experiments and reported the accuracy achieved; the lower the accuracy is, the more important the field is. We can see that in this case the most important fields are: *Title, Type* and *By.* For assessing the importance of the fields when using classifiers we conducted a chi-squared test and evaluated the information gain; the higher the score, the more important the field is. For both tests *Title, By* and *Abstract* were identified as the most important fields.

Table 3: Attribute selection.

Field	Leave-1-out	Chi-squared	Info gain
Title	0.737	671.510	0.352
Subtitle	0.790	0	0
Abstract	0.796	156.448	0.076
By	0.782	163.730	0.081
In	0.788	89.198	0.042
Type	0.781	0	0
Organization	0.793	2.665	0.001
Publisher	0.792	29.074	0.013

For all the different automatic methods the *Title* is the most discriminative field, but the next fields of secondary importance depend on the method. The Duplicate Scorer

considers *Type* more important than *By*, while the classifiers consider the *Type* information as being unimportant. The classifiers also consider the *Abstract* (which is actually the most discriminative field) as important, although it is not present in many of the publication pairs.

5.3 Peculiarities of Automatic Methods

The maximum optimized accuracy for the Duplicates Scorer that we can achieve on the ground truth is 0.83. This corresponds to the following learned weights: *Abstract=0.1, Title=0.9, Organization=0.1, Subtitle=0.1, By=0.3, Type=0.8, In=0.3, Publisher=0.0* and *Threshold=0.76*. This is consistent with the results obtained using the 'leave one field out' experiments in Section 5.2. The fields with the highest weights are the *Title*, *Type* and *By*.

The rules learned by the best performing classifier confirm the importance of the attributes that we examined in Section 5.2. When using the Jaccard similarity for both *c-gt* and *c-c* the learned tree first compares the *Title* similarity, then the *Type*, *Publisher* and finally the *By* similarities. For the *gt-gt* setting the classifier compares just first the *Title* and then the *Publisher*. When using the Needleman-Wunch similarity for both *c-gt* and *c-c*, the learned tree first compares the *Title* similarity, and then either the *Abstract* or *In* similarities. In the *gt-gt* setting the only rule is to compare the *Title* similarity.

5.4 Resource Allocation

The resources to be allocated for a round of active learning from the crowd, are the sample size s (the number of pairs from which we build the HITs to be posted on Mturk), and the number of assignments per HIT (corresponding to the number of workers that vote on a pair). In this section we explore the effects of different levels of resource allocation on the performance of the automatic method, the Duplicates Scorer after learning the optimal parameters, and the classifiers after retraining respectively.

5.4.1 How Many Assignments per Task?

We use the majority voting as a label aggregation strategy. This is similar to considering that all the workers have the same confidence. To determine the optimal number of votes we need for a pair to be accurately adjudged on MTurk, we experiment on a batch of 570 pairs having 7 votes and we compute the accuracy obtained by taking into consideration only the first workers that assigned labels on each pair. We plot the accuracy obtained by using different Automatic Methods for different number of assignments in Figure 5.

In the case of using the Duplicates Scorer, the performance we gain by introducing more than 3 workers is of only 1% for 7 workers and negligible when using 5 workers. Although the difference in accuracy between the setting where we use 1 worker and the setting with 3 workers is of 2%, we cannot use just 1 worker since on doing so we risk relying on only one opinion, which might be biased.

In the case of using classifiers we notice a drop in performance when using 5 workers when compared to using just 3 workers, that is recovered when using 7 workers for Naive Bayes and Decision Trees. For SVM the best performance is obtained when using just 1 worker, and using 3 causes a drop that is slowly recovered by adding additional workers.

For almost all the considered automatic methods, the additional cost of adding 2 or 4 more votes when we already have 3, is not justified by the performance increase. Therefore, 3 is optimal number of workers for the task of gathering labels for deduplication scientific publications.

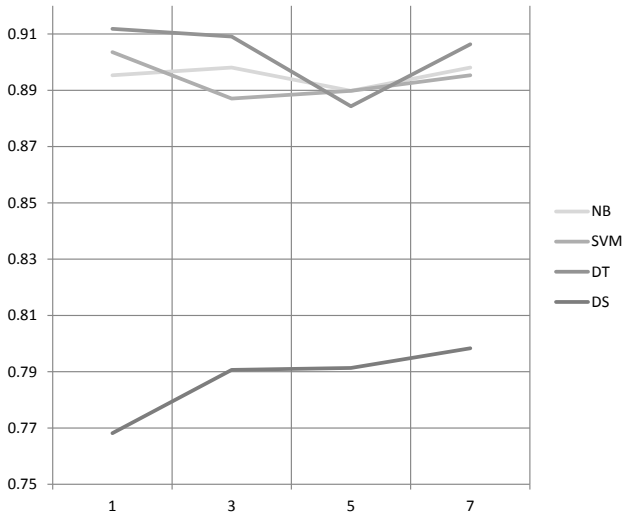


Figure 5: Number of assignments per task vs. accuracy of the automatic methods.

5.4.2 How Many Instances per Round?

To determine the optimal size of the sample that has to be labeled by the crowd in an active learning round, we experiment with 1070 pairs of training data obtained by using 3 workers. We regard these workers to be equally competent, and thereby the crowd decision is based on majority voting. According to this voting strategy 408 pair are duplicates and 662 are non-duplicates. We select samples of pairs of different sizes containing a number of duplicate or non-duplicate pairs proportional to the number of overall available pairs. We present the average accuracy obtained from 20 rounds of randomly sampling the pairs and retraining. We plot the accuracy obtained on optimizing for different number of pairs in Figure 6. For all the automatic methods we notice a considerable increase in performance for up to 500 examples per round; going over this number of examples does not result in an improvement that would justify the extra cost.

5.4.3 Eliminating Unreliable Workers

We investigate the effect of eliminating the workers that do not provide reliable assignments, on the accuracy of the Automatic Method. Instead of considering all workers as being equally good at deduplication, we compute the worker confidences using an iterative EM algorithm presented in [12]. Thus, the decision of different workers will be weighted by the confidence we have in him. We experiment with eliminating the workers whose computed confidence is under the reliability threshold, $WQThreshold$. The complete dataset containing all workers consists of 570 pairs, each having 7 votes from workers with different confidences. We optimize the parameters of the DuplicatesScorer using the pairs and votes that remain after eliminating the unreliable workers. We compare the effect of using different thresholds on the accuracy, obtained by comparing the crowd decision to the labels assigned by experts contained in our ground truth. For each threshold we compute the accuracy, show how many

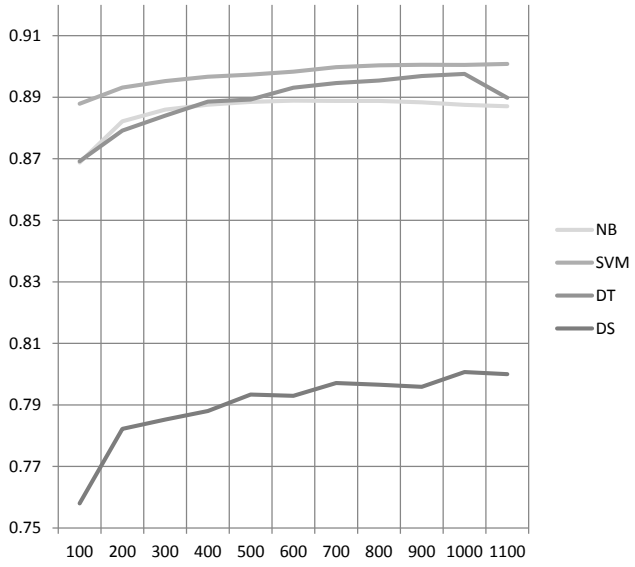


Figure 6: Number of tasks for each round vs. accuracy of the automatic methods.

Table 4: The distribution of the number available votes according to each selected worker quality threshold.

QT	Number of available votes						
	1	2	3	4	5	6	7
0.6	0	0	4	47	64	113	342
0.65	0	0	4	47	64	119	336
0.7	0	0	4	67	208	224	67
0.75	0	65	220	181	95	8	1
0.8	13	90	233	151	78	4	1
0.85	145	185	96	77	43	1	0
0.9	172	236	100	34	1	0	0
0.95	158	15	0	0	0	0	0
1	113	9	0	0	0	0	0

pairs are consequently left for optimization, and how many workers exhibit a confidence greater than the threshold, as presented in Table 5. We present the distribution of the number of votes for the available pairs in Table 4.

On examining Table 5 we notice that the maximum accuracy for the Duplicates Scorer is obtained when eliminating all the workers having a confidence under 0.9. This retains a reasonable number of pairs, based on which the optimization of the DuplicatesScorer can be carried out, while filtering out half of the workers. Although the number of pairs is consistent, by looking at Table 4 we see that most of these pairs have only 3 votes assigned to them. Comparing to the setting where we use just 3 votes, and do not eliminate any of the workers the gain in accuracy is of only 1%, but the costs are much higher, for obtaining the extra labels that will be discarded. Thus, in this setting using just 3 votes without eliminating the poorly performing workers, can be recommended. The workers identified as unreliable should only be prevented from participating in the next rounds.

For the classifiers, the best thresholds for eliminating workers $WQThreshold$ is quite low, 0.7 for Naive Bayes and the Support Vector Machine classifiers, while for the Decision Trees a threshold of 0.65 or even 0.6 seems to provide a good performance. The low threshold allows more workers to participate in the decision process for the aggregated crowd label. This is not considered by the classifiers as contributing noise, but as a means to improve their performance. The classifiers are more robust and relatively independent of the worker quality threshold, and at the same time they show superior performance when compared to the Duplicates Scorer.

Table 5: Statistics on the pairs samples after eliminating unreliable workers. The Accuracies obtained by different methods: Naive Bayes (NB), Support Vector Machines(SVM), Decision Tree(DT) classifiers and the Duplicates Scorer(DS), along with the Number of available pairs for training (P) and the number of workers(Wrk) contributing votes to those pairs, according to the selected worker quality threshold(QT).

QT	NB	SVM	DT	DS	P	Wrk
60	89.81	89.81	90.08	79.66	570	88
65	90.08	89.81	91.18	79.64	570	84
70	90.08	90.08	89.53	79.35	570	81
75	89.26	89.81	90.63	80.47	570	77
80	88.43	89.81	90.36	80.26	570	73
85	89.26	89.81	88.98	80.48	547	52
90	89.26	89.81	88.98	80.99	543	42
95	84.30	87.33	87.60	79.30	173	28
100	84.30	85.40	90.63	80.00	122	24

5.5 Candidates Selection Strategies

In this section we experiment with the entire dataset that was labeled by the crowd by simulating the label acquisition process. In each step we procure labels from the crowd for corresponding $s \in \{10, 20, 50\}$ instances and we apply our method. We have chosen to use three types of automatic methods that we want to improve: Duplicates Scorer and Naive Bayes using the Jaccard similarity or the Needleman-Wunch similarity. We plot the Accuracy obtained at each step, using different sampling strategies for the instances that are to be labeled. The different candidate selection strategies we use in our experiments are *Uncertain*, *Random*, and *Representative*.

The *Representative* strategy divides the pool of unlabeled instances into bins according to the uncertainty of their labels. Bin B_u contains instances with an uncertainty in the $[0.1 \cdot u, 0.1 \cdot (u+1)]$ interval, with $u \in \{0, 1, .., 9\}$, covering in this way the whole uncertainty interval $[0, 1]$. Thereafter, in each round of candidate selection it selects a number of random instances from each bin that is proportional to the binsize. In this way we do not choose the most uncertain instances, but instances with representative uncertainty.

If we use the Naive Bayes classifier as the automatic machine learning algorithm: *Uncertain* refers to getting the s pairs that have a probability to belong to one class closest to the uncertainty threshold (0.5). *Random* chooses s pairs randomly from the entire dataset. For the *Representative* strategy the measure of uncertainty is computed as the difference between the probability of belonging to the negative class and the probability of belonging to the positive class. We experiment with both similarity metrics for computing the features: Jaccard and Needleman-Wunch.

In the case of the Duplicates Scorer as the automatic algorithm that we propose to use in our method, *Uncertain* refers to getting the s pairs that have a ASL that is closest

to the learned threshold for that step. *Random* chooses s pairs randomly from the entire dataset, independent of the ASL For the *Representative* strategy, the ASL is the used measure of uncertainty.

We report the accuracy of the automatic method after retraining for each round of active learning in terms of learning curves in Figure 7. For the classifier *Random* performs better than the *Uncertain* strategy, pointing to the fact that a strategy that takes into account the representativeness of the instances would yield better results. Using the *Uncertain* strategy, the crucial example that improves the performance significantly is discovered later than in the *Random* case. Of course we notice an improvement of the performance as more instances become available with each round of label gathering. As expected, this example is found early by the *Representative* selection strategy. The *Representative* strategy performs better than the *Uncertain* strategy, but it is not clearly better than the *Random* sampling.

In the case of using the Duplicates Scorer as the automatic algorithm that learns from the crowd, we see that using 10 or 20 pairs is not efficient for both *Random* and *Uncertain*. We do not see an increase in performance with each round as we would expect. This is different for 50 examples per round, we see the increase in performance, and also the *Uncertain* strategy beats the *Random* sampling. We can conclude that 10 and 20 are a small quantity of new labels for our method to exhibit improvement. This effect can only be seen from upwards of 50 instances. In the resource allocation experiments presented in Section 5.4, we observe that a good number of instances per round is around 500.

6. DISCUSSION

6.1 Agreement Influence

As noticed in Section 5.1 the performance of the automatic method is dependent on the training and testing data. We assume this has something to do with the difference in quality that can also be noticed in the agreement between the labelers as reported in Section 4.2. In the *c-c*, we observe that when the testing data has the aggregated crowd labels as ground truth, the performance is much lower than in the other cases. The agreement on the crowd data when using 3 labels in the experiment in terms of Fleiss' κ and in terms of Krippendorfs's α are portrayed in Table 2. This is much lower than the agreement of the experts on the ground truth data that is used, respectively. By comparing the agreement of the experts with the agreement of the crowd labelers, we observe that the crowd labelers agree less, maybe because of their lack of domain knowledge.

One other concern that is raised regarding the agreement between labelers, is related to the instances that are retained in the loop until the aggregated label surpasses a desired level of confidence. There might be some instances for which assigning a label is a hard task, on which the different assessors cannot reach an agreement. These instances would remain in the loop for a long time and waste resources in vain. Our method would benefit from a way to identify instances that are hard to label. These kinds of instances could be directed towards expert labelers instead of the crowd.

6.2 Comparison of Automatic Methods

We can notice a clear inferiority of the Duplicates Scorer when compared to employing classifiers. The Duplicate Scorer was developed for a setting where the attributes are unknown a priori. It was intended to be deployed in case neither the type of entity nor the attribute labels to be used in order to describe the entities are known. In such scenarios, machine learning methods cannot be used, because the features are unknown. The threshold can however be learned, as shown in [12]. For the scenario in the current paper, where the attributes (features) are known and fixed, the best approach seems to be employing classifiers. The advantage of the Duplicates Scorer is that it is applicable to unknown entities, at the cost of a decrease in accuracy.

Selecting an appropriate method that learns from the crowd is very important, because it influences the candidates selection strategy. We cannot use a selection strategy based on one method to train another method. As each new label has an associated cost, the right method and the right selection strategy both play a key role in achieving the desired performance within the required budget and time.

6.3 Crowd vs. Experts

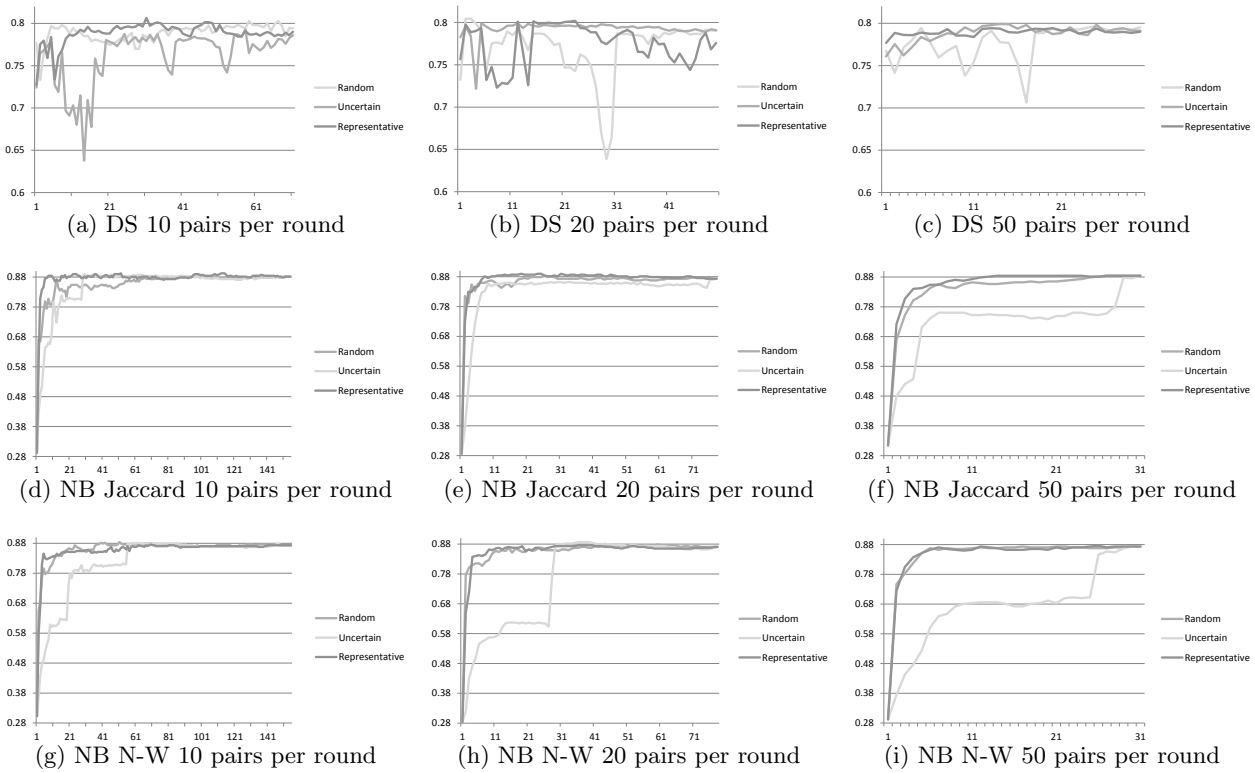
The results of crowdsourcing are strongly influenced by the difficulty and complexity of the task. Although the task we want to solve in the present paper is easy at first glance, being reduced to comparing different fields, it is not a trivial task and even the experts had disagreements. We have discovered that it is not as trivial for crowd workers as it is for people with domain knowledge. This is evident when we look into the agreement between the crowd workers and compare it to the agreement between experts.

Examining Table 2 we can notice that the agreement between the crowd workers is much lower than the agreement between experts. We have also investigated the most common reasons why the crowd workers assign the wrong labels, or where they disagree the most. It leads us to the conclusion that the crowd workers, not possessing enough domain knowledge, oversimplify the task, and only compare the titles of the publications. They completely ignore the other fields, and they superficially compare the titles. Domain knowledge such as recognizing the abbreviations of conference titles, or the patience the experts have proven to exhibit are crucial in assigning the correct labels.

6.4 Data Distribution

If we examine the label distribution in the ground truth we can notice a high unbalance towards pairs that are not duplicates. Thus just by assigning the non-duplicate label, we could achieve an accuracy of about 70%. If we consider majority voting as the label aggregation strategy, the crowd data is also unbalanced consisting of 61% non-duplicate pairs and 39% duplicate pairs. In this particular scenario, detecting duplicates in a publication database, this is not unusual, reflecting the real situation where the number of duplicates depends on the number of sources and their quality.

The field distribution presented in Section 4 could also offer some insight into the importance of the attributes, or how the crowd would handle the work assigned to it. The fields that appear in both publications of most pairs are: *Title*, *By*, *In* and *Type*. The other fields are either both empty or one of them is empty. As expected, those fields that are mostly empty for both publications of a pair are identified as the most unimportant in the attribute selection experiments and also have the lowest weight for the learned Duplicate Scorer. These fields are not mentioned at all in the



(a) DS 10 pairs per round (b) DS 20 pairs per round (c) DS 50 pairs per round

(d) NB Jaccard 10 pairs per round (e) NB Jaccard 20 pairs per round (f) NB Jaccard 50 pairs per round

(g) NB N-W 10 pairs per round (h) NB N-W 20 pairs per round (i) NB N-W 50 pairs per round

Figure 7: Learning curves for active learning on crowd data collected using different automatic methods, candidate instance selection strategies and number of instances for each active learning round.

rules for the trained Decision Trees. The *Title* is identified by all the methods as being the most discriminative field, but that is only because the *Abstract* is not present in all the pairs. The importance of the other fields that can be taken into consideration when drawing a comparison depends on the automatic method we use and how it learns.

7. CONCLUSIONS AND FUTURE WORK

In this paper we investigated a methodology for employing crowdsourcing in active learning, and identified some of the challenges that arise. We underline the importance of choosing an appropriate automatic method that learns from the crowd as this choice directly influences the selection of candidate instances.

In our investigations regarding the allocation of resources we have discovered evidence that there exists a certain threshold over which spending more money either in terms of the number of workers or the number of tasks per active learning round does not give a proportional increase in quality. The quality seems to rise as we allocate more resources, but after a certain point in seems to plateau. This is task-related and we do not claim that the limits we discovered in our experiments are universal. Carrying out such experiments when employing methods like this in order to determine the right level of resource allocation, therefore helping in keeping within the budget constraint are recommended. The selection strategy plays an important role in how fast the automatic method learns from the crowd. In our experiments we have noticed that a representative strategy is superior to the uncertain strategy as mentioned in the literature.

The proposed framework for performing active learning with crowdsourcing is complex and has many components that rely on each other to function well. We have experimentally tested some of the components and their interplay in a well defined scenario, and we believe that the proposed methods can be successfully applied in other domains for other tasks similarly.

For future work we consider experiments using other types of tasks and data, using and comparing different label aggregation strategies. We intend to delve further into the influence of agreement on the quality of the labels and of the trained methods. We plan to look more closely into advanced worker quality assessments, taking into account the temporal dimension, and trying to recover those workers that consistently introduce errors. As a continuation to the task on which we experimentally apply our methods we could enhance the duplicates identification pipeline by to learning from the crowd how to create a unified representation of all the detected duplicates to be displayed in a search engine, in order not to clutter the results list.

A feedback mechanism for workers could be employed such that the automatic method can support the worker when he commits a mistake and attempt to correct him so that the humans and machines truly work together learning from one another. For this we would need worker authentication, in order to control the assignment process. The controlled assignment will allow us to use only the best workers or the workers that seem to learn and improve over time. This is currently not available in Mechanical Turk but we believe it will set the precedent in the future.

Acknowledgments

This work was partially supported by the CUbRIK and DURAARK projects funded by the European Commission under the 7^{th} Framework Programme (Contract No. 287704 and 600908).

8. REFERENCES

[1] V. Ambati, S. Hewavitharana, S. Vogel, and J. Carbonell. Active learning with multiple annotations for comparable data classification task. In *Proceedings of the 4th Workshop on Building and Using Comparable Corpora: Comparable Corpora and the Web*. Association for Computational Linguistics, 2011.

[2] V. Ambati, S. Vogel, and J. G. Carbonell. Active learning and crowd-sourcing for machine translation. In *LREC*, volume 11. Citeseer, 2010.

[3] J. Attenberg and F. Provost. Why label when you can search?: alternatives to active learning for applying human resources to build classification models under extreme class imbalance. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10. ACM, 2010.

[4] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S. E. Whang, and J. Widom. Swoosh: a generic approach to entity resolution. *The VLDB Journal*, 18(1), Jan. 2009.

[5] M. Bilenko, R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg. Adaptive Name Matching in Information Integration. *IEEE Intelligent Systems*, 18(5), 2003.

[6] E. Chatzilari, S. Nikolopoulos, Y. Kompatsiaris, and J. Kittler. Active learning in social context for image classification. In *9th International Conference on Computer Vision Theory and Applications*, VISAPP, 2014.

[7] D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Mach. Learn.*, 15(2), 1994.

[8] A. P. Dawid and A. M. Skene. Maximum Likelihood Estimation of Observer Error-Rates Using the EM Algorithm. *Applied Statistics*, 28(1), 1979.

[9] G. Demartini, D. E. Difallah, and P. Cudré-Mauroux. Zencrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12. ACM, 2012.

[10] A. Doan, Y. Lu, Y. Lee, and J. Han. Object Matching for Information Integration: A Profiler-Based Approach. In *IIWeb*, 2003.

[11] P. Donmez and J. G. Carbonell. Proactive learning: cost-sensitive active learning with multiple imperfect oracles. In *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM, 2008.

[12] M. Georgescu, D. D. Pham, C. S. Firan, W. Nejdl, and J. Gaugaz. Map to humans and reduce error: crowdsourcing for deduplication applied to digital libraries. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, CIKM '12. ACM, 2012.

[13] A. Y. Halevy, X. Dong, J. Madhavan, A. Y. Halevy, and J. Madhavan. Reference reconciliation in complex information spaces. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data - SIGMOD '05*. ACM Press, 2005.

[14] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1), 2009.

[15] M. A. Hernández and S. J. Stolfo. Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem. *Data Min. Knowl. Discov.*, 2(1), 1998.

[16] E. Ioannou, C. Niederée, and W. Nejdl. Probabilistic Entity Linkage for Heterogeneous Information Spaces. In *CAiSE*, 2008.

[17] P. G. Ipeirotis, F. Provost, and J. Wang. Quality management on amazon mechanical turk. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, HCOMP '10. ACM, 2010.

[18] F. Laws, C. Scheible, and H. Schütze. Active learning with amazon mechanical turk. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2011.

[19] M. Lease. On quality control and machine learning in crowdsourcing. In *Human Computation*, 2011.

[20] Z. Miklós, N. Bonvin, P. Bouquet, M. Catasta, D. Cordioli, P. Fankhauser, J. Gaugaz, E. Ioannou, H. Koshutanski, A. Maña, C. Niederée, T. Palpanas, and H. Stoermer. From Web Data to Entities and Back. *CAiSE*, 2010.

[21] A. Morris, Y. Velegrakis, and P. Bouquet. Entity Identification on the Semantic Web. In *SWAP*, 2008.

[22] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy. Learning from crowds. *Journal of Machine Learning Research*, 11, 2010.

[23] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *KDD*, 2002.

[24] B. Settles. Active learning literature survey. *University of Wisconsin, Madison*, 2010.

[25] V. S. Sheng, F. Provost, and P. G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08. ACM, 2008.

[26] A. Sheshadri and M. Lease. Square: A benchmark for research on computing crowd consensus. In *First AAAI Conference on Human Computation and Crowdsourcing*, 2013.

[27] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics, 2008.

[28] O. Tamuz, C. Liu, S. Belongie, O. Shamir, A. Kalai, and A. Kalai. Adaptively learning the crowd kernel. In *ICML*, 2011.

[29] S. Vijayanarasimhan and K. Grauman. Large-scale live active learning: Training object detectors with crawled data and crowds. In *Computer Vision and Pattern Recognition*, CVPR. IEEE, 2011.

[30] L. von Ahn. Human computation. In *CIVR*, 2009.

[31] J. Wang, T. Kraska, M. J. Franklin, and J. Feng. Crowder: Crowdsourcing entity resolution. *Proceedings of the VLDB Endowment*, 5(11), 2012.

[32] P. Welinder, S. Branson, P. Perona, and S. J. Belongie. The multidimensional wisdom of crowds. In *Advances in Neural Information Processing Systems*, 2010.

[33] J. Whitehill, T.-f. Wu, J. Bergsma, J. R. Movellan, and P. L. Ruvolo. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in neural information processing systems*, 2009.

[34] Y. Yan, G. M. Fung, R. Rosales, and J. G. Dy. Active learning from crowds. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011.

[35] H. Yang, A. Mityagin, K. M. Svore, and S. Markov. Collecting high quality overlapping labels at low cost. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2010.

[36] M.-C. Yuen, L.-J. Chen, I. King, and I. King. A survey of human computation systems. In *CSE (4)*, 2009.

[37] L. Zhao, G. Sukthankar, and R. Sukthankar. Incremental relabeling for active learning with noisy crowdsourced annotations. In *Privacy, security, risk and trust (passat), 2011 ieee third international conference on and 2011 ieee third international conference on social computing (socialcom)*. IEEE, 2011.